US010176048B2

(12) United States Patent
Jaquette et al.

(10) Patent No.: US 10,176,048 B2
(45) Date of Patent: Jan. 8, 2019

(54) CREATING A RESTORE COPY FROM A COPY OF SOURCE DATA IN A REPOSITORY HAVING SOURCE DATA AT DIFFERENT POINT-IN-TIMES AND READING DATA FROM THE REPOSITORY FOR THE RESTORE COPY

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(72) Inventors: **Glen A. Jaquette**, Tucson, AZ (US); **Gregory T. Kishi**, Oro Valley, AZ (US)

(73) Assignee: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 53 days.

(21) Appl. No.: **14/175,987**

(22) Filed: **Feb. 7, 2014**

(65) **Prior Publication Data**

US 2015/0227433 A1 Aug. 13, 2015

(51) **Int. Cl.**
G06F 7/00 (2006.01)
G06F 11/14 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 11/1448*** (2013.01); ***G06F 11/1469*** (2013.01); *G06F 2201/84* (2013.01)

(58) **Field of Classification Search**
CPC G06F 11/1448; G06F 17/30864; G06F 2201/84; G06F 11/1446

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,594,744 B1 * 7/2003 Humlicek G06F 3/0601 707/999.202
6,912,629 B1 * 6/2005 West G06F 11/1456 711/161

(Continued)

FOREIGN PATENT DOCUMENTS

CN 1071770 A 5/1993
CN 1740981 3/2006

(Continued)

OTHER PUBLICATIONS

PCT International Search Report and Written Opinion dated Sep. 6, 2015 for Application No. PCT/IB2015/052543 filed Apr. 8, 2015, pp. 10.

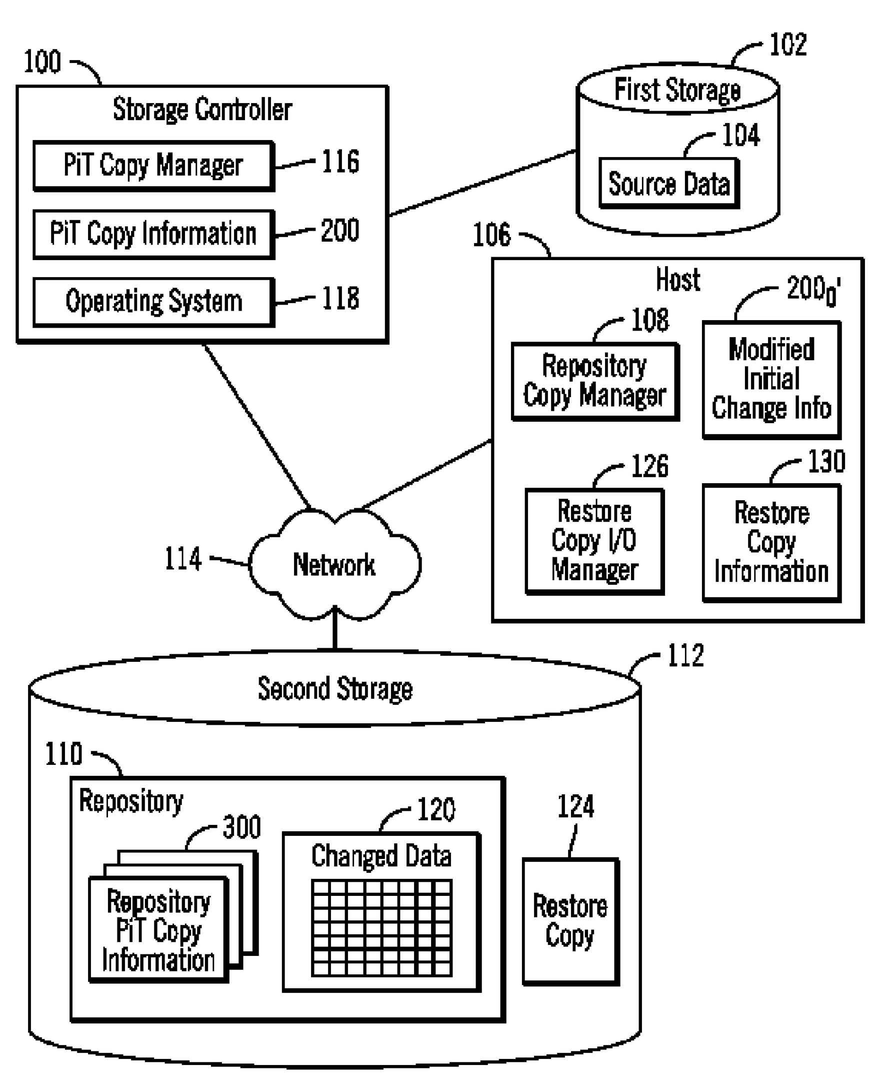
(Continued)

Primary Examiner — Mohammed R Uddin
(74) *Attorney, Agent, or Firm* — David W. Victor; Konrad Raynes Davda & Victor LLP

(57) **ABSTRACT**

Provided are a computer program product, system, and method for creating a restore copy from a copy of source data in a repository having source data at different point-in-times and reading data from the repository for the restore copy. The source data is copied as of an initial point-in-time to a repository. Point-in-time copies at different point-in-times of the source data are initiated following the initial point-in-time. Change information for the point-in-time copy indicating changed data in the source data that changed between the point-in-time of the point-in-time copy and a subsequent point-in-time are transmitted to the repository. For each point-in-time copy, changed source data indicated in the change information for the point-in-time copy as changed is copied to the repository. A restore copy is returned to a restore request before the source data in the

(Continued)



repository as of a restore point-in-time is copied to the restore copy.

24 Claims, 11 Drawing Sheets

(58) **Field of Classification Search**
USPC 707/646, 649, 655, 999.204
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,107,486	B2	9/2006	Okada et al.	
7,133,884	B1 *	11/2006	Murley	G06F 17/30348
7,284,104	B1 *	10/2007	Wu	G06F 11/1451 707/999.202
7,318,134	B1	1/2008	Oliveira et al.	
7,529,782	B2 *	5/2009	Prahlad	G06F 11/1435
7,650,533	B1 *	1/2010	Saxena	G06F 11/1451 714/13
7,809,691	B1 *	10/2010	Karmarkar	G06F 11/1469 707/674
7,822,717	B2 *	10/2010	Kapoor	G06F 11/1469 707/680
7,836,267	B1	11/2010	Cross	
7,934,064	B1	4/2011	Per et al.	
8,060,713	B1	11/2011	Natanzon	
8,095,804	B1	1/2012	Armangau et al.	
8,099,572	B1	1/2012	Arora et al.	
8,166,128	B1	4/2012	Faulkner et al.	
8,250,033	B1	8/2012	De Souter et al.	
8,260,752	B1	9/2012	Stringham et al.	
8,352,422	B2	1/2013	Prahlad et al.	
8,375,158	B2	2/2013	Cho	
8,380,939	B2	2/2013	Agesen	
8,386,733	B1 *	2/2013	Tsaur	G06F 11/1458 711/162
8,417,872	B2	4/2013	Bae et al.	
8,453,145	B1	5/2013	Naik	
8,473,697	B1	6/2013	Tiwari et al.	
8,495,316	B2 *	7/2013	Nagarkar	G06F 9/45533 711/162
8,555,009	B1 *	10/2013	Majahan	G06F 11/1446 711/162
8,566,510	B2	10/2013	Weingarten	
8,666,944	B2	3/2014	Beatty et al.	
8,738,883	B2	5/2014	Wade et al.	
8,788,769	B2	7/2014	Abercrombie et al.	
8,818,936	B1 *	8/2014	Haase	G06F 17/30115 707/610
8,819,362	B1	8/2014	Duprey et al.	
8,904,125	B1	12/2014	Elling et al.	
2003/0167380	A1	9/2003	Green et al.	
2004/0117572	A1	6/2004	Welsh et al.	
2004/0158766	A1 *	8/2004	Liccione	G06F 11/2023 714/4.11
2004/0186968	A1	9/2004	Factor et al.	
2004/0260726	A1 *	12/2004	Hrle	G06F 11/1466
2004/0260895	A1	12/2004	Werner et al.	
2005/0108302	A1 *	5/2005	Rand	G06F 11/1469
2005/0210320	A1	9/2005	Vincent	
2005/0216788	A1 *	9/2005	Mani-Meitav	G06F 11/1456 714/6.32
2006/0041727	A1	2/2006	Adkins et al.	
2006/0173935	A1	8/2006	Merchant et al.	
2006/0224636	A1	10/2006	Kathuria et al.	
2007/0022319	A1 *	1/2007	Haselton	G06F 11/1451 714/15
2007/0055833	A1	3/2007	Vu et al.	
2007/0276885	A1 *	11/2007	Valiyaparambil	G06F 11/1451
2007/0277010	A1 *	11/2007	Anand	G06F 11/1458 711/162
2008/0098195	A1	4/2008	Cheon et al.	
2008/0162590	A1	7/2008	Kundu et al.	
2008/0243954	A1	10/2008	Augenstein et al.	
2009/0150626	A1 *	6/2009	Benhase	G06F 11/1456 711/162
2009/0193064	A1	7/2009	Chen et al.	
2009/0307286	A1	12/2009	Laffin	
2010/0049929	A1	2/2010	Nagarkar et al.	
2011/0093436	A1 *	4/2011	Zha	G06F 17/30592 707/639
2011/0173404	A1	7/2011	Eastman et al.	
2011/0191295	A1 *	8/2011	Ozdemir	G06F 17/30 707/639
2012/0078855	A1	3/2012	Beatty et al.	
2012/0130956	A1	5/2012	Caputo	
2012/0136832	A1	5/2012	Sadhwani	
2012/0158662	A1 *	6/2012	Buragohain	G06F 11/1461 707/649
2012/0197842	A1 *	8/2012	Marie	G06F 17/30088 707/639
2012/0278539	A1	11/2012	Hosogaya	
2012/0284232	A1	11/2012	Fiske	
2013/0006944	A1	1/2013	Prahlad et al.	
2013/0042054	A1	2/2013	Jung et al.	
2013/0046734	A1	2/2013	Cisler et al.	
2013/0103649	A9	4/2013	Linde	
2013/0132346	A1	5/2013	Varadarajan	
2013/0173552	A1	7/2013	Marie et al.	
2013/0318318	A1	11/2013	Nemoto et al.	
2013/0326125	A1	12/2013	Chang et al.	
2014/0052693	A1	2/2014	Zha et al.	
2014/0095823	A1	4/2014	Shaikh et al.	
2014/0149695	A1	5/2014	Zaslavsky et al.	
2014/0164660	A1	6/2014	DeCesaris et al.	
2014/0258241	A1	9/2014	Chen et al.	
2014/0258613	A1	9/2014	Sampathkumar	

FOREIGN PATENT DOCUMENTS

CN	101149694		3/2008
CN	102236589		4/2013
CN	103262043		8/2013
EP	2234018		3/2012
WO	9308529	A1	4/1993
WO	0101257		6/2001
WO	2010065271		8/2010
WO	2013175422		11/2013

OTHER PUBLICATIONS

Office Action dated Dec. 30, 2015, pp. 36, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.

Chesarek, et al., "IBM System Storage FlashCopy Manager and PPRC Manager Overview", IBM Corporation, Document No. REDP-4065, pp. 62, 2008.

Burger, et al., IBM System Storage DS8000 Series: IBM FlashCopy SE, IBM Corporation, Document No. REDP-4368-00, pp. 80, Feb. 2008.

Cronauer, et al., "IBM System Storage DS8000 Copy Services for Open Systems", IBM Corporation, Document No. SG24-6788-06, pp. 800, Feb. 2013.

Using a Repository Having a Full Copy of Source Data and Point-In-Time Information From Point-In-Time Copies of the Source Data to Restore the Source Data at Different Points-In-Time, by G.A. Jaquette et al., US Application Serial No. Unknown, filed Feb. 7, 2014.

Creating a Restore Copy From a Copy of a Full Copy of Source Data in a Repository That is at a Different Point-In-Time Than a Restore Point-In-Time of a Restore Request, by G.A. Jaquette et al., US Application Serial No. Unknown, filed Feb. 7, 2014.

Creating a Restore Copy From a Copy of Source Data in a Repository Having Source Data at Different Point-In-Times, by G.A. Jaquette et al., US Application Serial Unknown, filed Feb. 7, 2014.

PCT International Search Report and Written Opinion dated Jun. 15, 2015, for Application No. PCT/IB2015/050225, filed Jan. 12, 2015.

(56) **References Cited**

OTHER PUBLICATIONS

U.S. Pat. No. 8,666,944 is the English language counterpart of Chinese Patent No. 103262043.
U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
English language translation of CN 1740981 filed Mar. 1, 2006.
English language translation of CN 102236589 filed Apr. 17, 2013.
English language translation of CN 103262043 filed Aug. 21, 2013.
Response dated Jun. 9, 2016, pp. 14, to Office Action dated Mar. 9, 2016, pp. 49, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Office Action dated Mar. 9, 2016, pp. 49, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Response dated Mar. 30, 2016, pp. 15, to Office Action dated Dec. 30, 2015, pp. 36, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Office Action dated May 20, 2016, pp. 52, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Final Office Action dated Jul. 15, 2016, pp. 40, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Final Office Action dated Sep. 23, 2016, pp. 35, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Office Action dated Jul. 21, 2016, pp. 52, for U.S. Appl. No. 14/175,980, filed Feb. 7, 2014.
Response dated Aug. 22, 2016, pp. 17, to Office Action dated May 20, 2016, pp. 52, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Response dated Oct. 21, 2016, pp. 14, for U.S. Appl. No. 14/175,980, filed Feb. 7, 2014.
Response dated Oct. 17, 2016, pp. 14, to Final Office Action dated Jul. 15, 2016, pp. 40, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Final Office Action dated Sep. 22, 2016, pp. 33, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
UK Examination Report dated Dec. 16, 2016, pp. 3, for Application No. 1619298.1, filed Apr. 8, 2015.
Response dated Dec. 20, 2016, pp. 15, to Office Action dated May 20, 2016, pp. 52, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
UK Reply to Examination Report dated Feb. 1, 2017, pp. 13, to UK Examination Report dated Dec. 16, 2016, pp. 3, for Application No. 1619298.1, filed Apr. 8, 2015.
Response dated Jan. 23, 2017, pp. 14, to Final Office Action dated Sep. 23, 2016, pp. 35, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Final Office Action dated Feb. 10, 2017, pp. 29, for U.S. Appl. No. 14/175,980, filed Feb. 7, 2014.
Advisory Action dated Jan. 23, 2017, pp. 8, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Preliminary Amendment dated Jan. 24, 2017, pp. 12, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Amendment dated May 10, 2017, pp. 15, to Final Office Action dated Feb. 10, 2017, pp. 29, for U.S. Appl. No. 14/175,980, filed Feb. 7, 2014.
Office Action dated Aug. 11, 2017, pp. 38, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Amendment dated Nov. 15, 2016, pp. 14, to Final Office Action dated Jul. 15, 2016, pp. 40, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Office Action dated Sep. 14, 2017, pp. 48, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Notice of Allowance dated Sep. 12, 2017, pp. 21, for U.S. Appl. No. 14/175,980, filed Feb. 7, 2014.
Office Action dated Sep. 14, 2017, pp. 41, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Response dated Dec. 14, 2017, pp. 15, to Office Action dated Sep. 14, 2017, pp. 48, for U.S. Appl. No. 14/175,964, filed Feb. 7, 2014.
Response dated Nov. 13, 2017, pp. 16, to Office Action dated Aug. 11, 2017, pp. 38, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Response dated Dec. 14, 2017, pp. 17, to Office Action dated Sep. 14, 2017, pp. 41, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Final Office Action dated Feb. 20, 2018, pp. 32, for U.S. Appl. No. 14/263,917, filed Apr. 28, 2014.
Final Office Action dated Feb. 23, 2018, pp. 37, for U.S. Appl. No. 14/175,975, filed Feb. 7, 2014.
Robocopy.exe, Microsoft, Corporation, Copyright 1985-2003, pp. 35.
Office Action dated Aug. 10, 2018, pp. 15, for U.S. Appl. No. 14/263,917.
Pre-Appeal Brief Request dated Jul. 30, 2018, pp. 6, for U.S. Appl. No. 14/175,964.
CN Office Action dated Jul. 3, 2018, pp. 7, for Application No. 201580007367.
English Machine Translation CN101149594, published Mar. 26, 2008.
Pre Appeal Brief dated Jun. 25, 2018 for U.S. Appl. No. 14/175,975, 6 Pgs.
Pre Appeal Brief dated May 21, 2018, for U.S. Appl. No. 14/263,917, 6 Pgs.
English Machine translation of CN Office Action dated Jul. 3, 2018, pp. 7, for Application No. 201580007367.
English Machine translation of CN1071770A, pp. 5, published.
WO9308529A1 published Apr. 29, 1993, pp. 34, is the English counterpart of CN101149694A.
English Machine translation of "Notification_of_Reasons_for_Refusal" for JP Application No. 2016-548352, dated Sep. 18, 2018, pp. 4.
"Notification_of_Reasons_for_Refusal" for JP Application No. 2016-548352, dated Sep. 18, 2018, pp. 6.
English machine translation of "Search Report by Registered Searching Organization", JP Application No. 2016-548352, dated Sep. 18, 2018, pp. 34.
"Search Report by Registered Searching Organization", dated Sep. 18, 2018, JP Application No. 2016-548352, pp. 29.
Information Materials for IDS, Sep. 21, 2018, pp. 1.

* cited by examiner

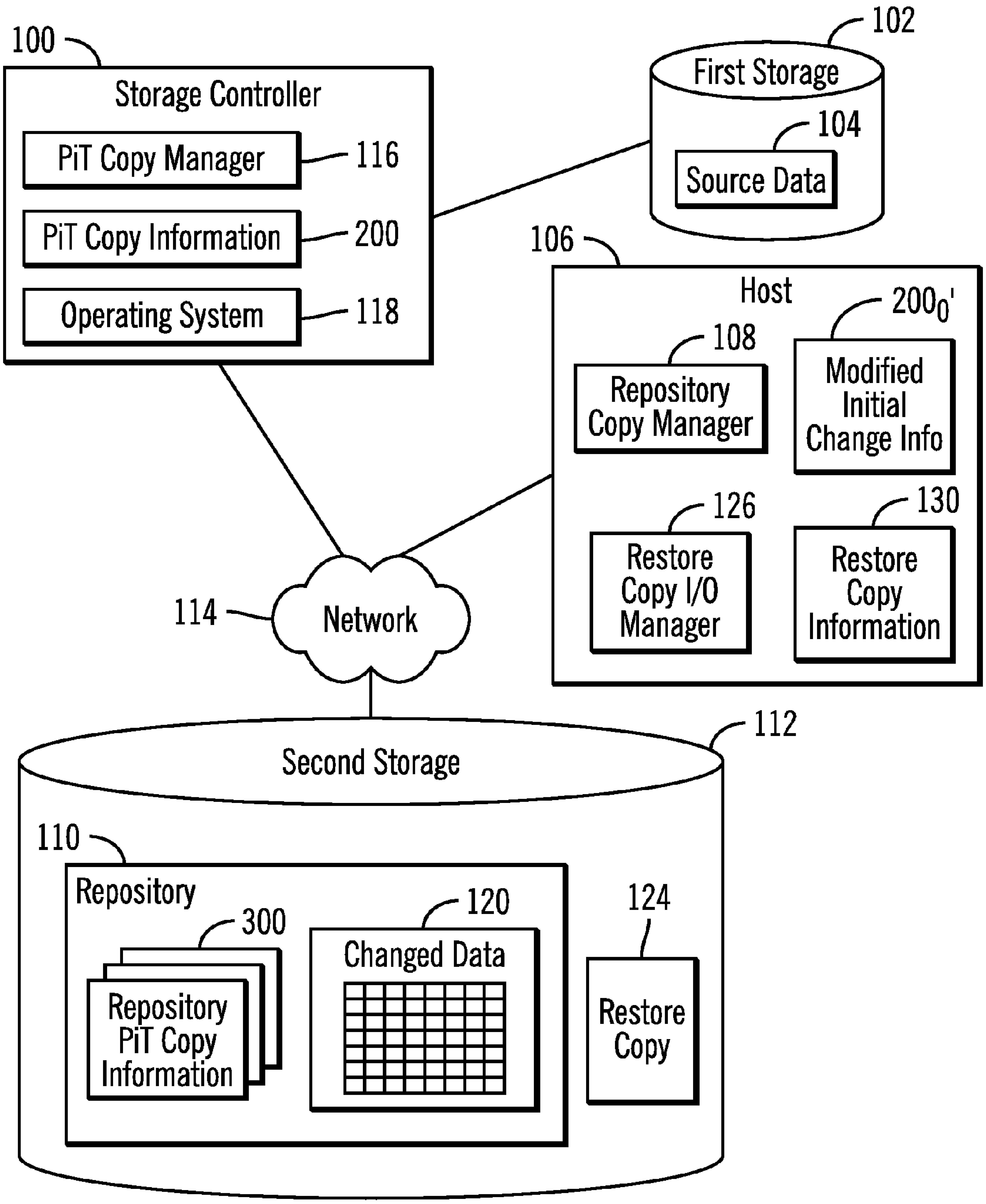

100
Storage Controller
PiT Copy Manager
116
PiT Copy Information
200
Operating System
118
102
First Storage
104
Source Data
106
Host
108
Repository Copy Manager
200_0'
Modified Initial Change Info
126
Restore Copy I/O Manager
130
Restore Copy Information
114
Network
112
Second Storage
110
Repository
300
Repository PiT Copy Information
120
Changed Data
124
Restore Copy

FIG. 1

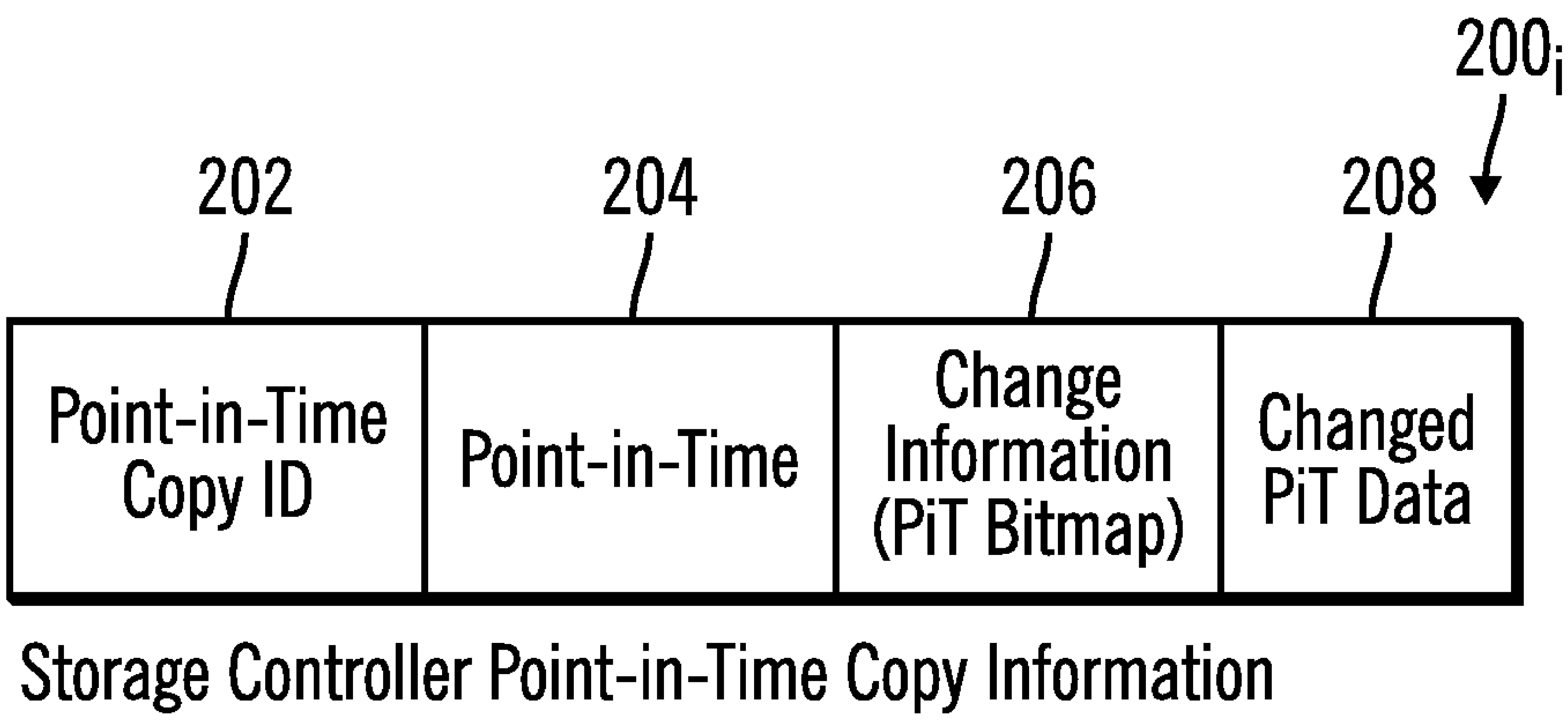

200i
202
204
206
208
Point-in-Time Copy ID
Point-in-Time
Change Information (PiT Bitmap)
Changed PiT Data
Storage Controller Point-in-Time Copy Information

FIG. 2

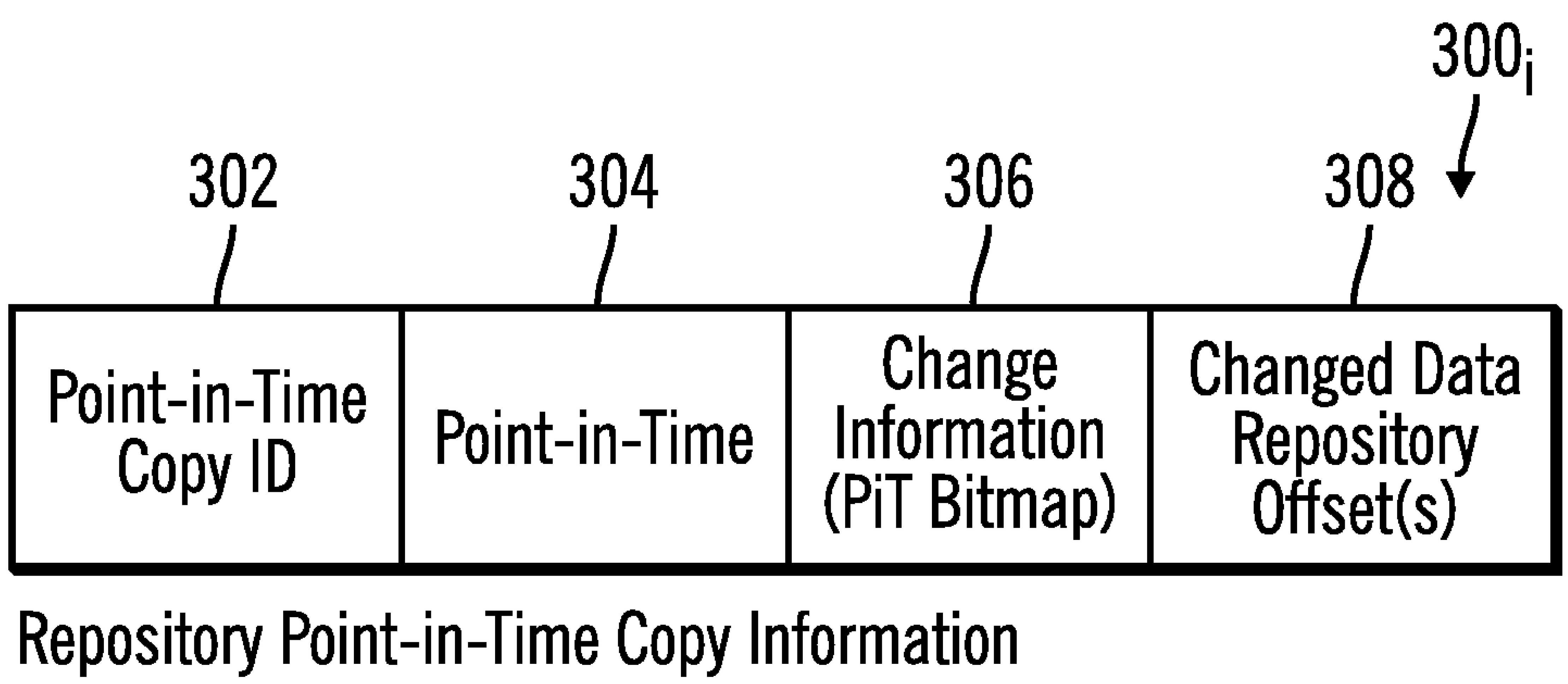

300i
302
304
306
308
Point-in-Time Copy ID
Point-in-Time
Change Information (PiT Bitmap)
Changed Data Repository Offset(s)
Repository Point-in-Time Copy Information

FIG. 3

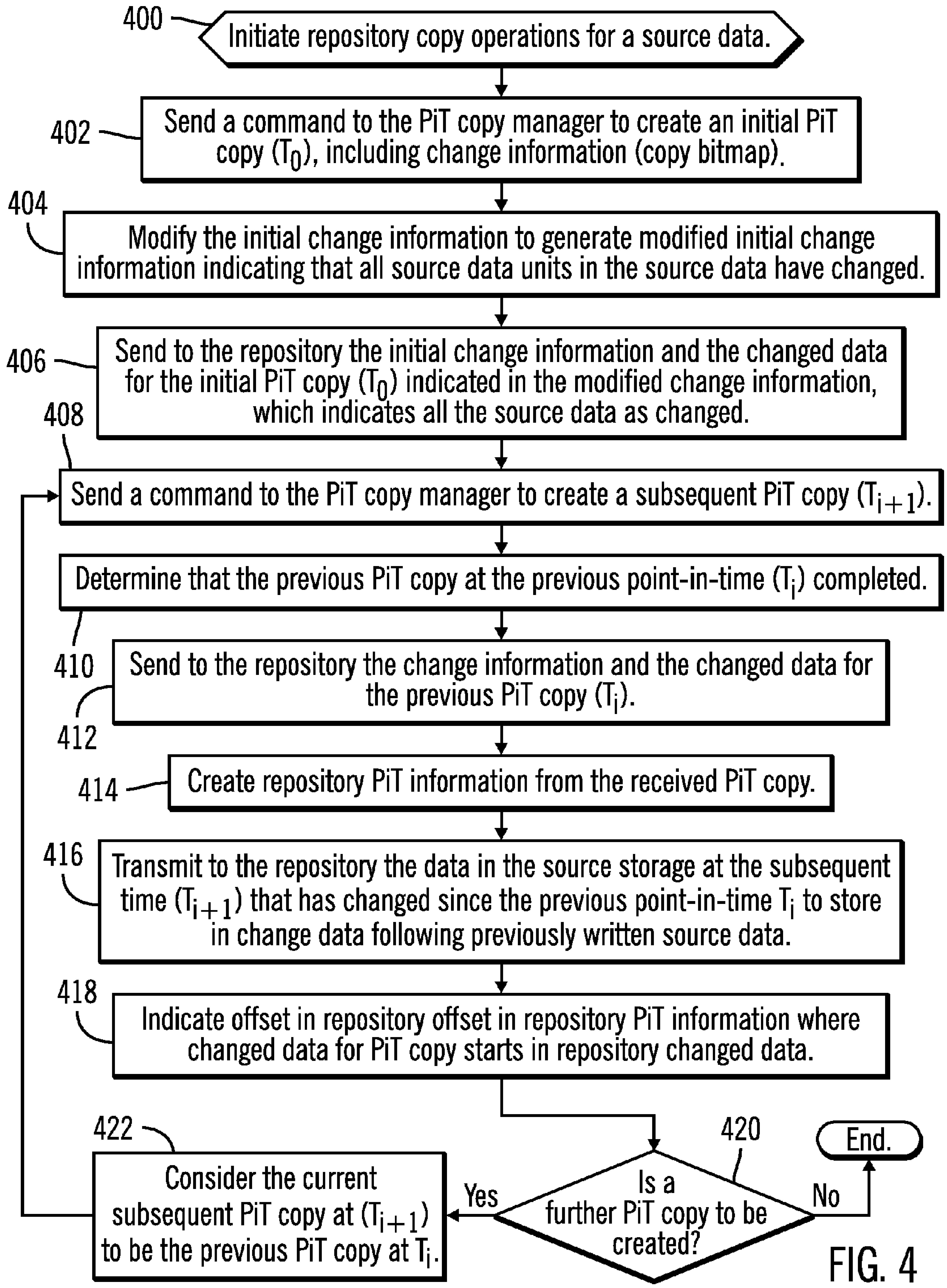
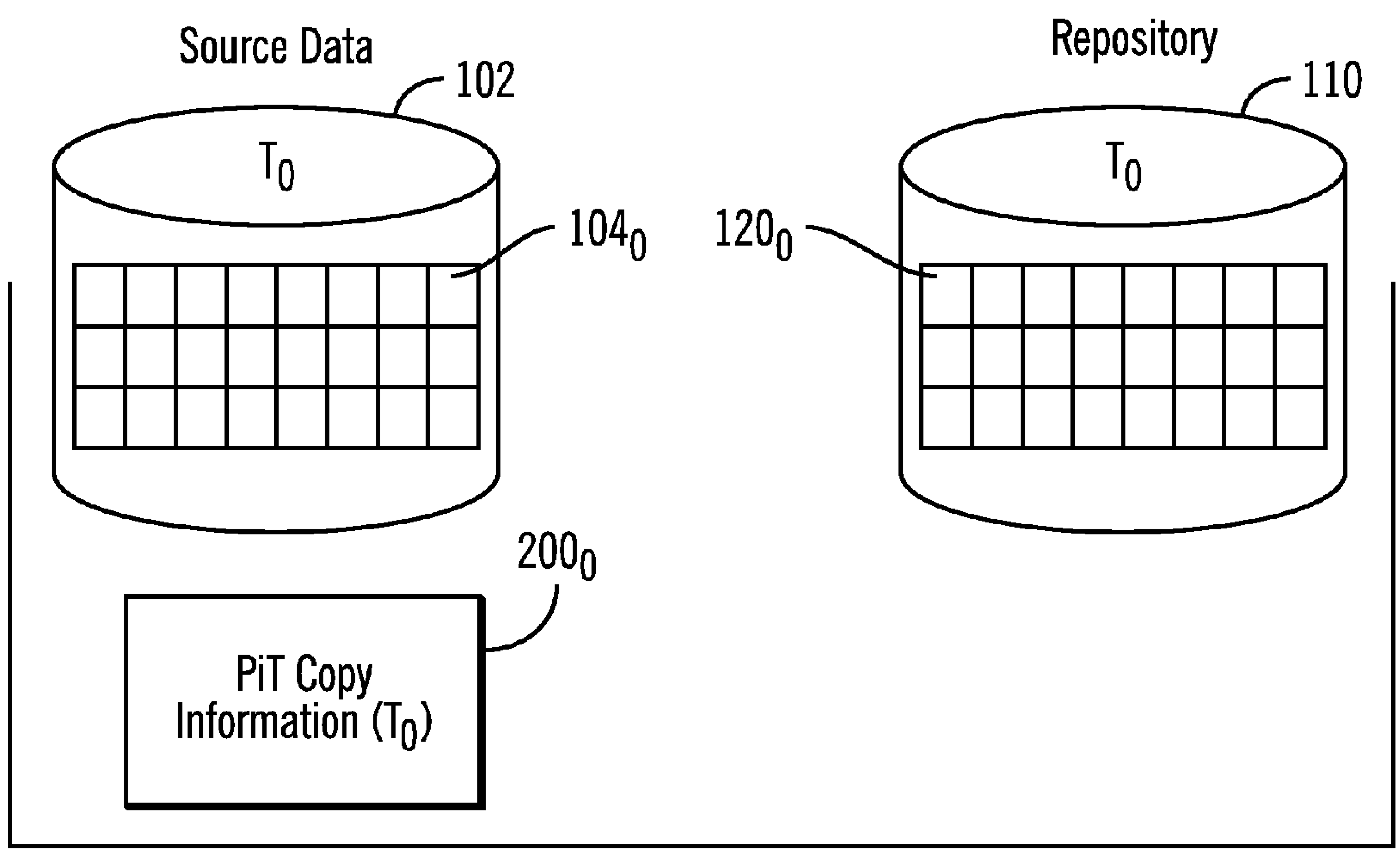
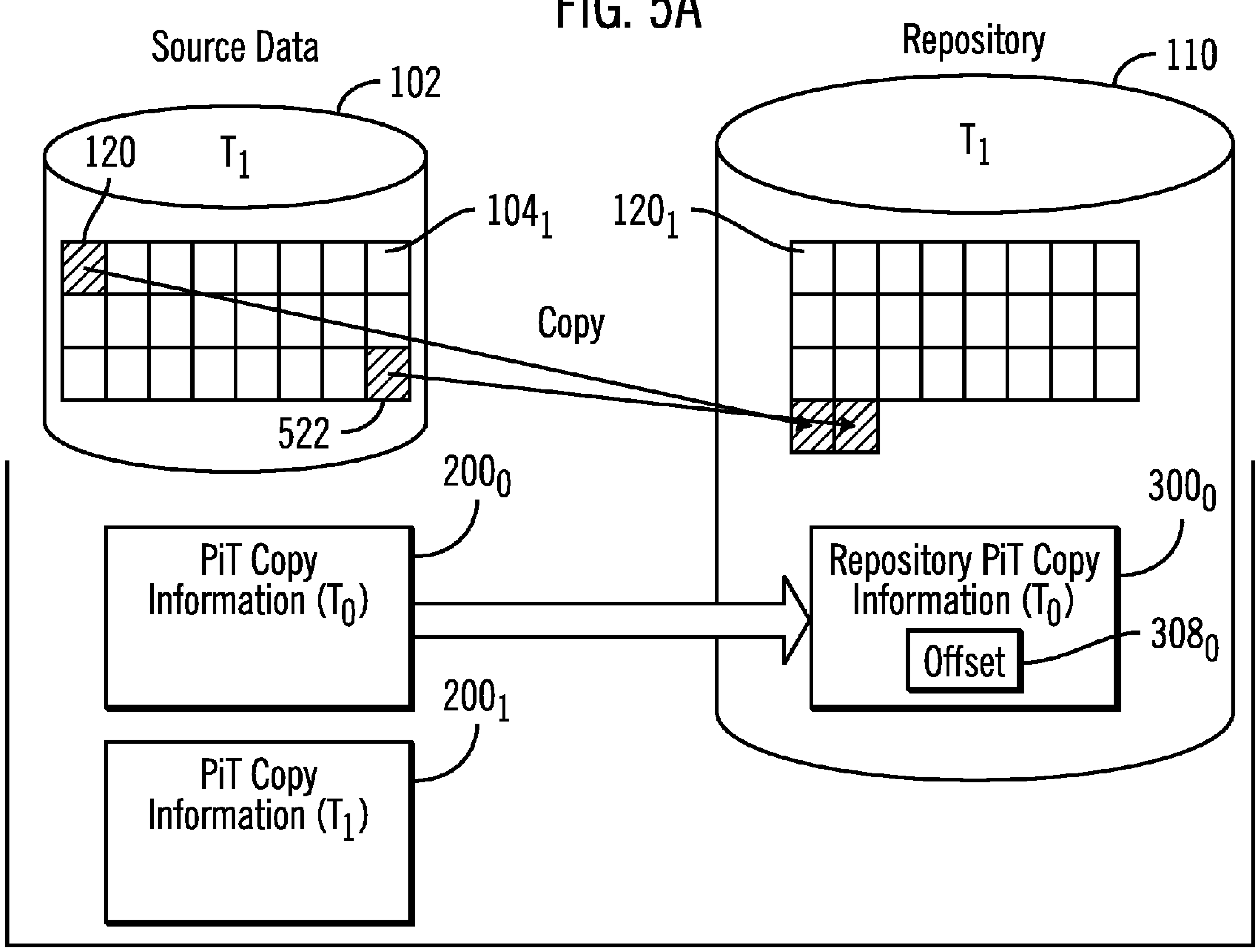

400
Initiate repository copy operations for a source data.
402
Send a command to the PiT copy manager to create an initial PiT copy (T0), including change information (copy bitmap).
404
Modify the initial change information to generate modified initial change information indicating that all source data units in the source data have changed.
406
Send to the repository the initial change information and the changed data for the initial PiT copy (T0) indicated in the modified change information, which indicates all the source data as changed.
408
Send a command to the PiT copy manager to create a subsequent PiT copy (Ti+1).
Determine that the previous PiT copy at the previous point-in-time (Ti) completed.
410
Send to the repository the change information and the changed data for the previous PiT copy (Ti).
412
414
Create repository PiT information from the received PiT copy.
416
Transmit to the repository the data in the source storage at the subsequent time (Ti+1) that has changed since the previous point-in-time Ti to store in change data following previously written source data.
418
Indicate offset in repository offset in repository PiT information where changed data for PiT copy starts in repository changed data.
422
Consider the current subsequent PiT copy at (Ti+1) to be the previous PiT copy at Ti.
Yes
420
Is a further PiT copy to be created?
No
End.

FIG. 4



Source Data
102
T_0
104_0
120_0
Repository
110
T_0
200_0
PiT Copy Information (T_0)

FIG. 5A



Source Data
102
120
T_1
104_1
120_1
Repository
110
T_1
Copy
522
200_0
PiT Copy Information (T_0)
300_0
Repository PiT Copy Information (T_0)
Offset
308_0
200_1
PiT Copy Information (T_1)

FIG. 5B

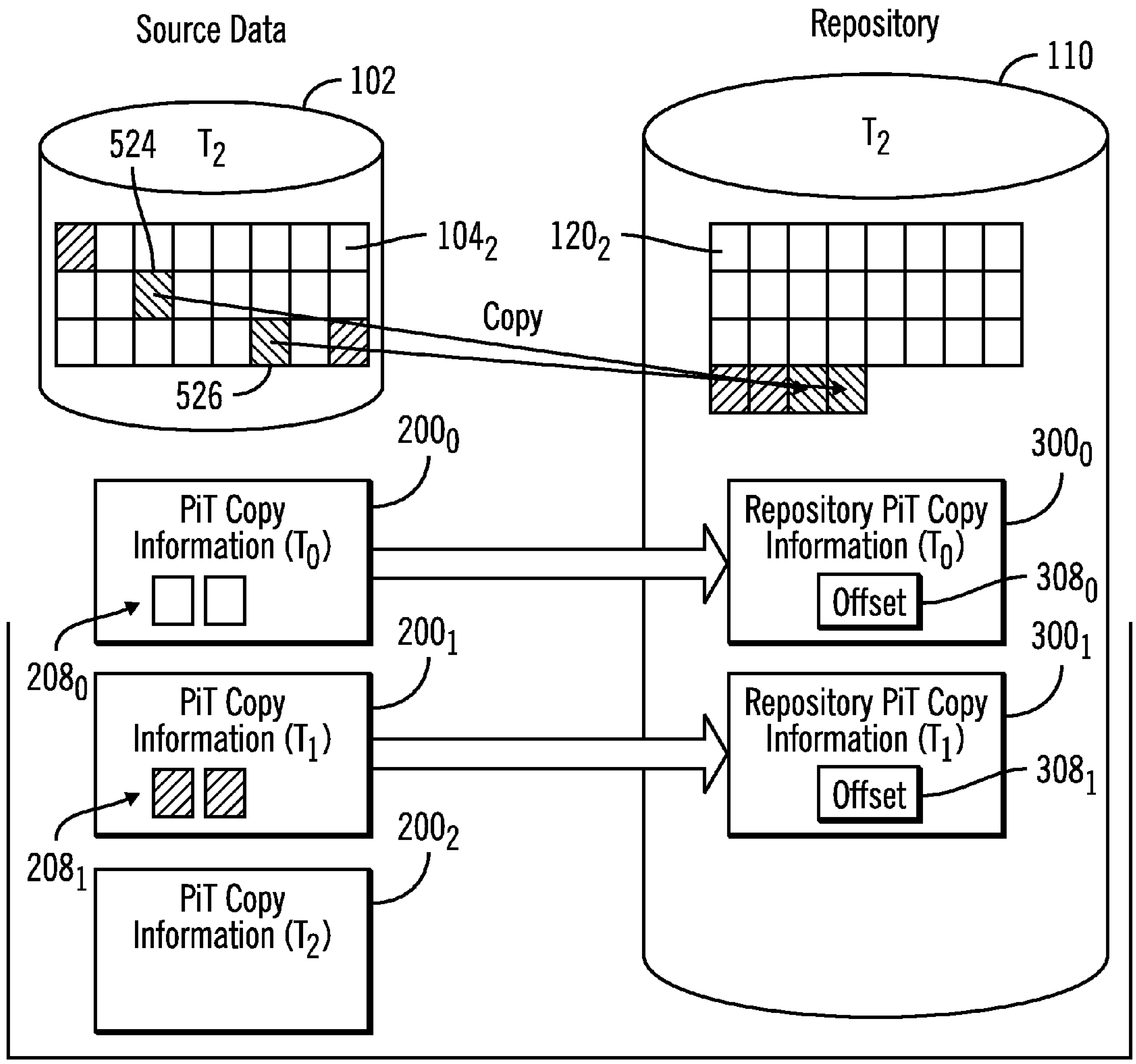

Source Data
102
524
T2
104₂
526
Copy
Repository
110
T2
120₂
200₀
PiT Copy Information (T0)
208₀
200₁
PiT Copy Information (T1)
208₁
200₂
PiT Copy Information (T2)
300₀
Repository PiT Copy Information (T0)
Offset
308₀
300₁
Repository PiT Copy Information (T1)
Offset
308₁

FIG. 5C

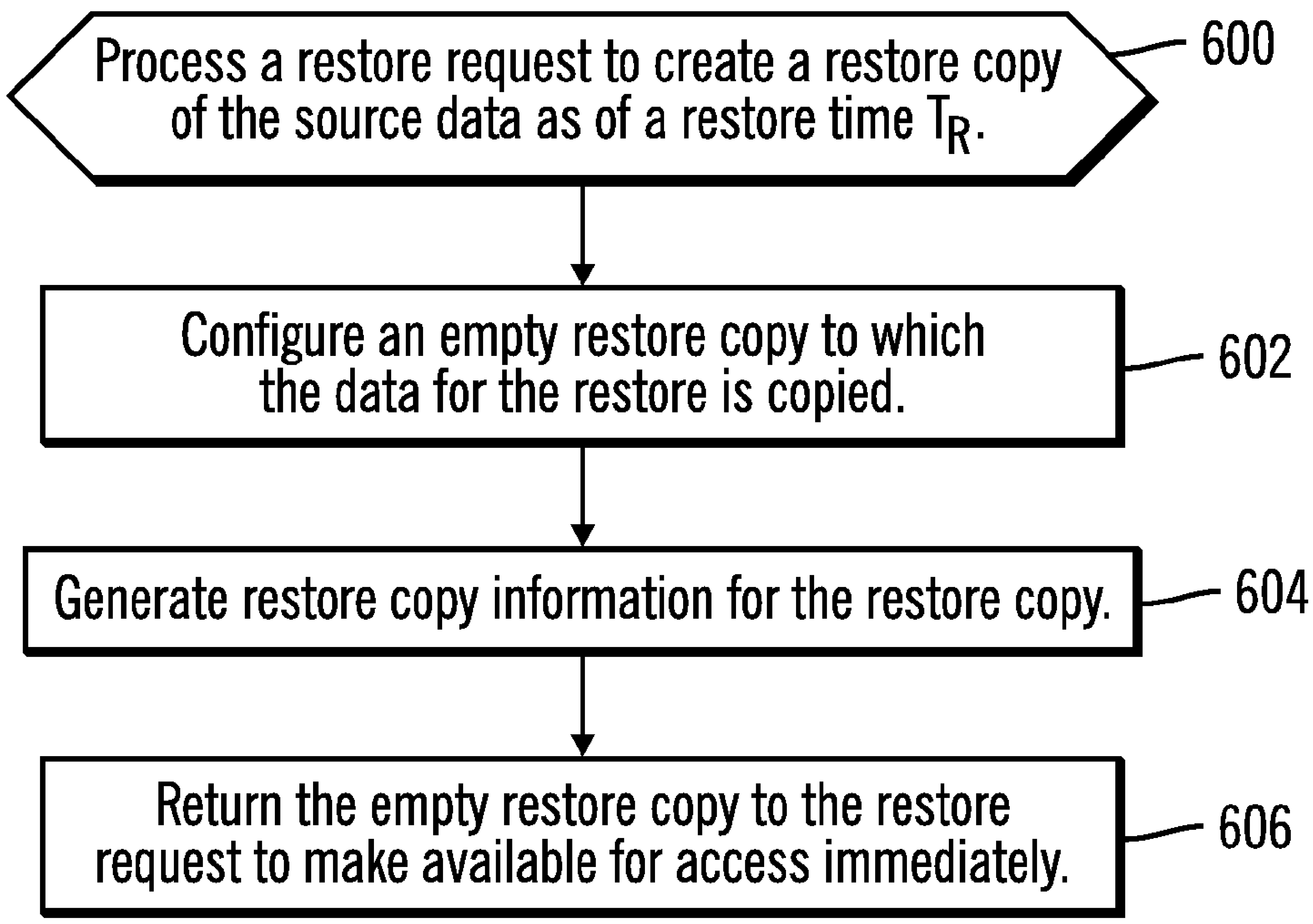
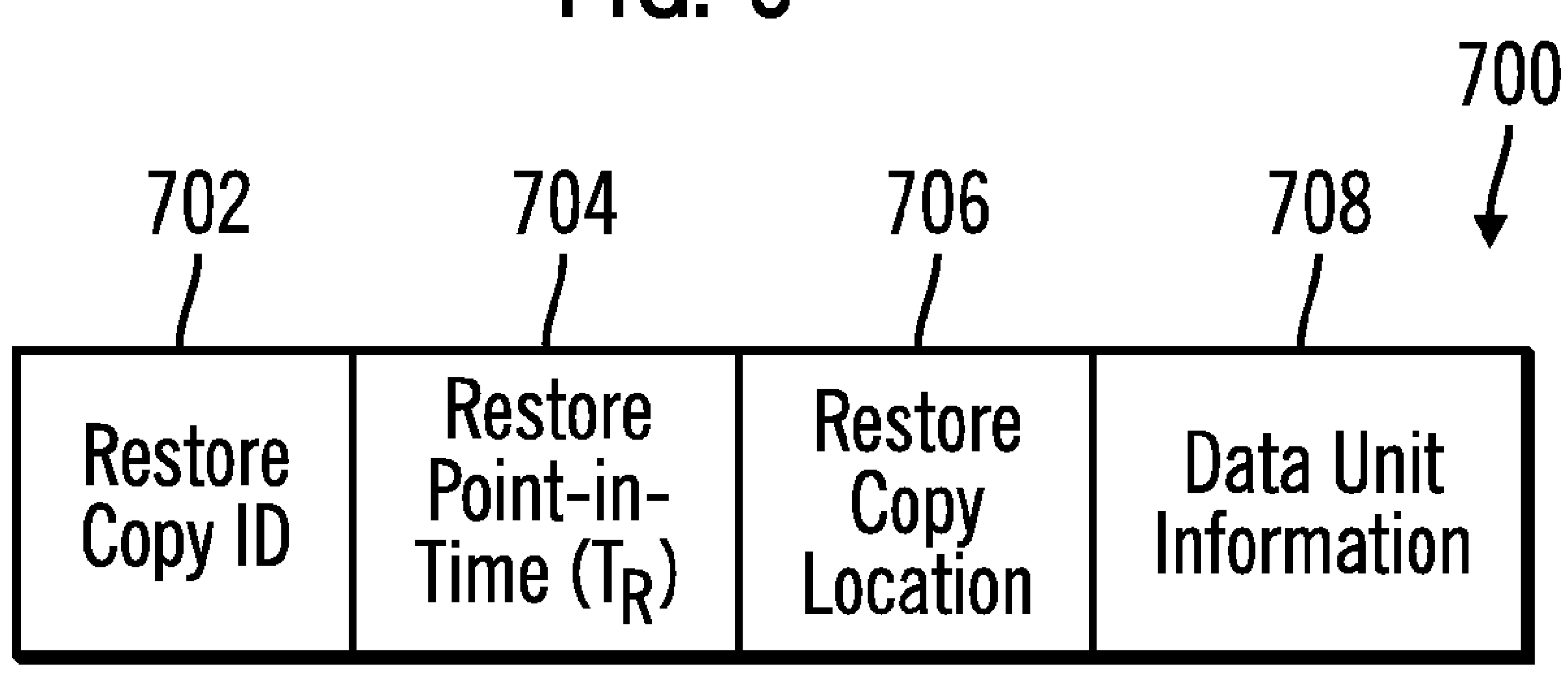
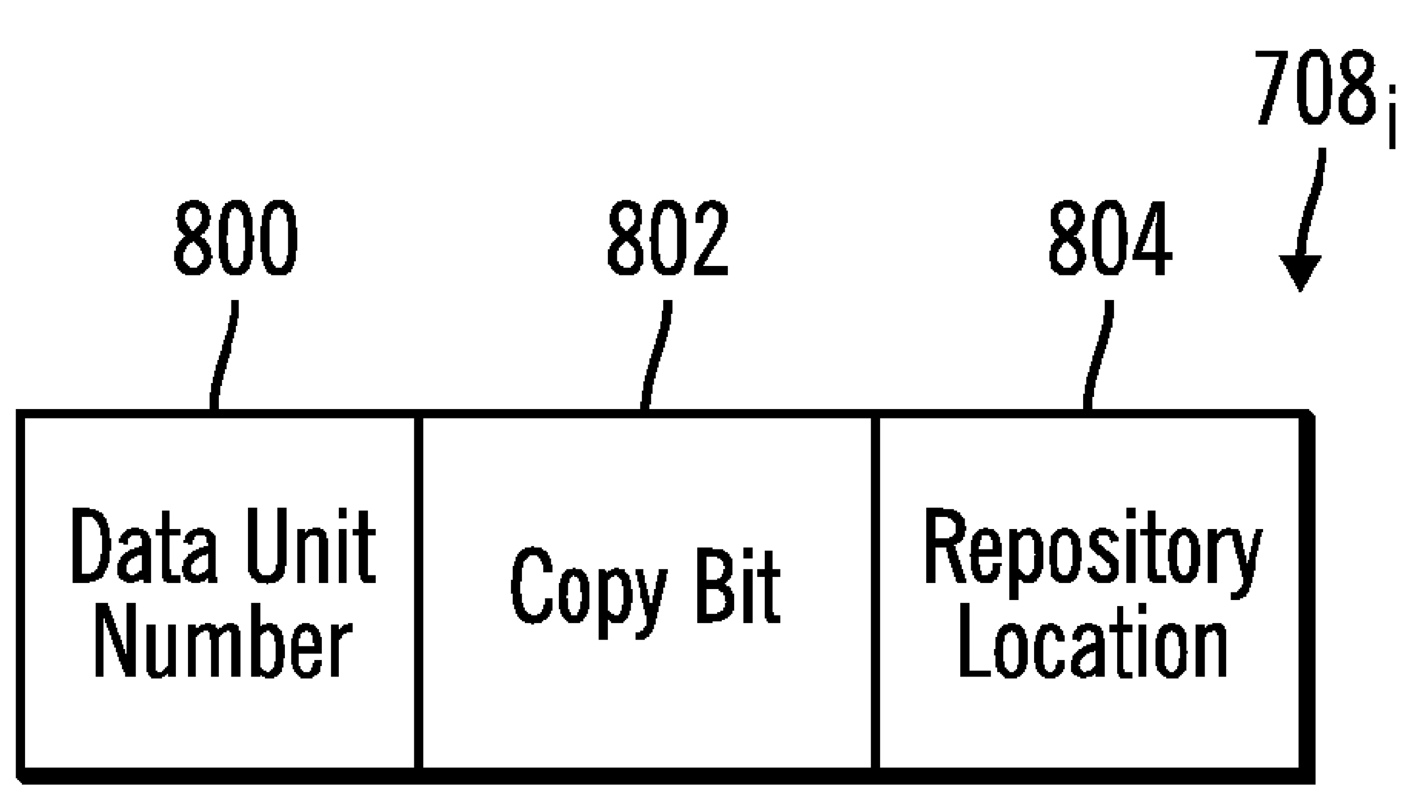


FIG. 6



Restore Copy Information

FIG. 7



Data Unit Information

FIG. 8

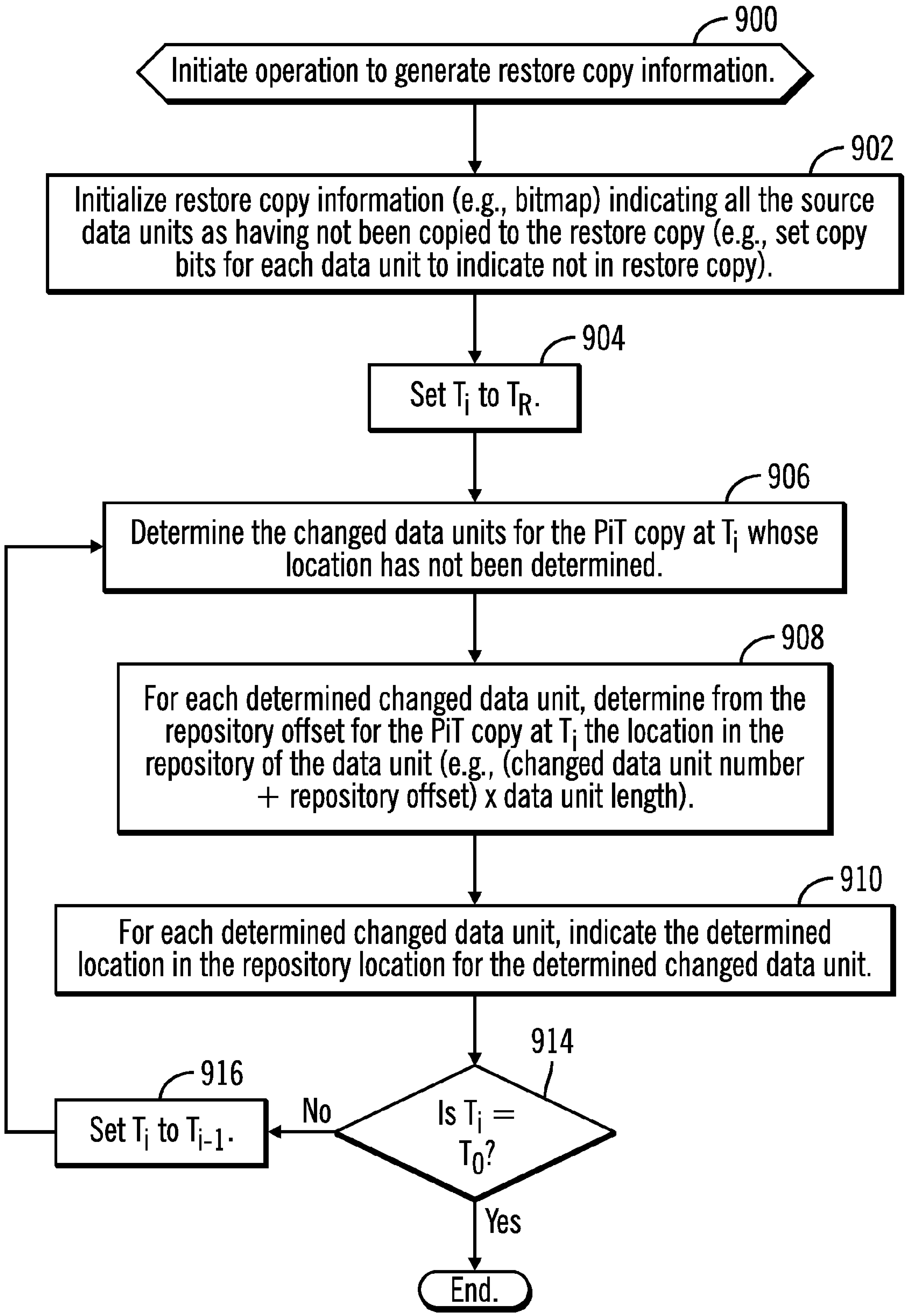

900
Initiate operation to generate restore copy information.
902
Initialize restore copy information (e.g., bitmap) indicating all the source data units as having not been copied to the restore copy (e.g., set copy bits for each data unit to indicate not in restore copy).
904
Set T_i to T_R.
906
Determine the changed data units for the PiT copy at T_i whose location has not been determined.
908
For each determined changed data unit, determine from the repository offset for the PiT copy at T_i the location in the repository of the data unit (e.g., (changed data unit number + repository offset) x data unit length).
910
For each determined changed data unit, indicate the determined location in the repository location for the determined changed data unit.
914
Is T_i = T_0?
No
916
Set T_i to T_{i-1}.
Yes
End.

FIG. 9

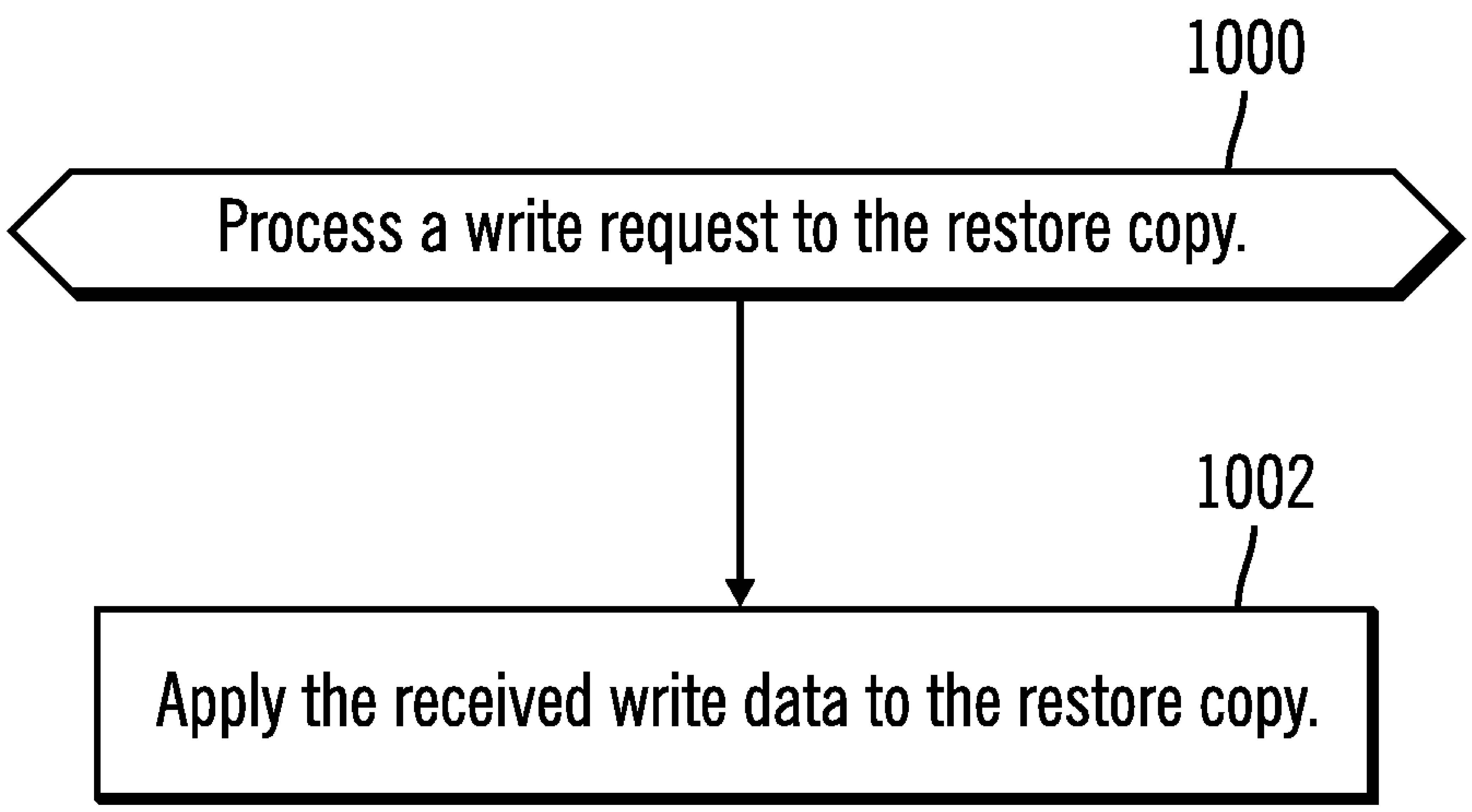

1000
Process a write request to the restore copy.
1002
Apply the received write data to the restore copy.

FIG. 10

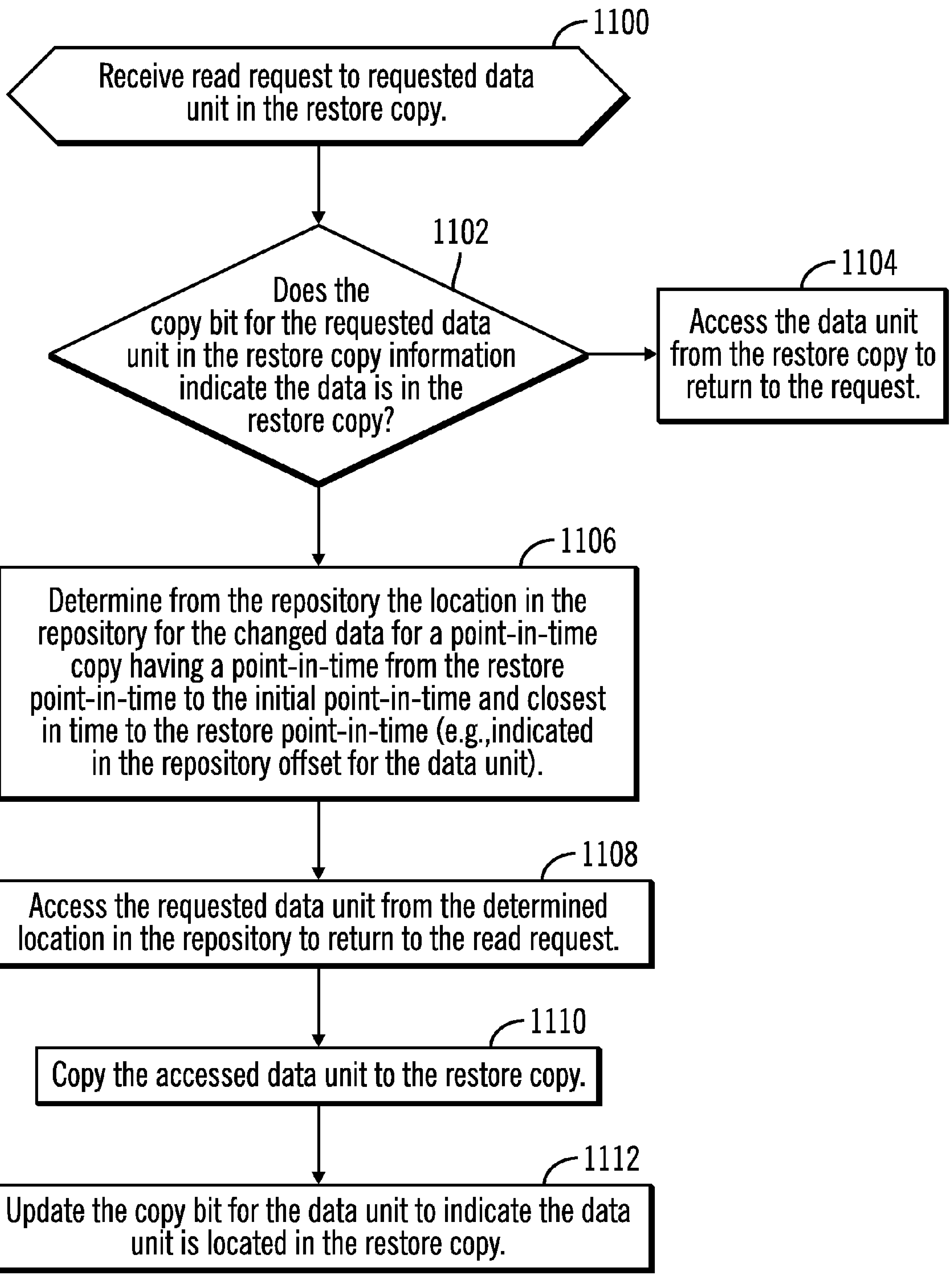

1100
Receive read request to requested data unit in the restore copy.
1102
Does the copy bit for the requested data unit in the restore copy information indicate the data is in the restore copy?
1104
Access the data unit from the restore copy to return to the request.
1106
Determine from the repository the location in the repository for the changed data for a point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time (e.g.,indicated in the repository offset for the data unit).
1108
Access the requested data unit from the determined location in the repository to return to the read request.
1110
Copy the accessed data unit to the restore copy.
1112
Update the copy bit for the data unit to indicate the data unit is located in the restore copy.

FIG. 11

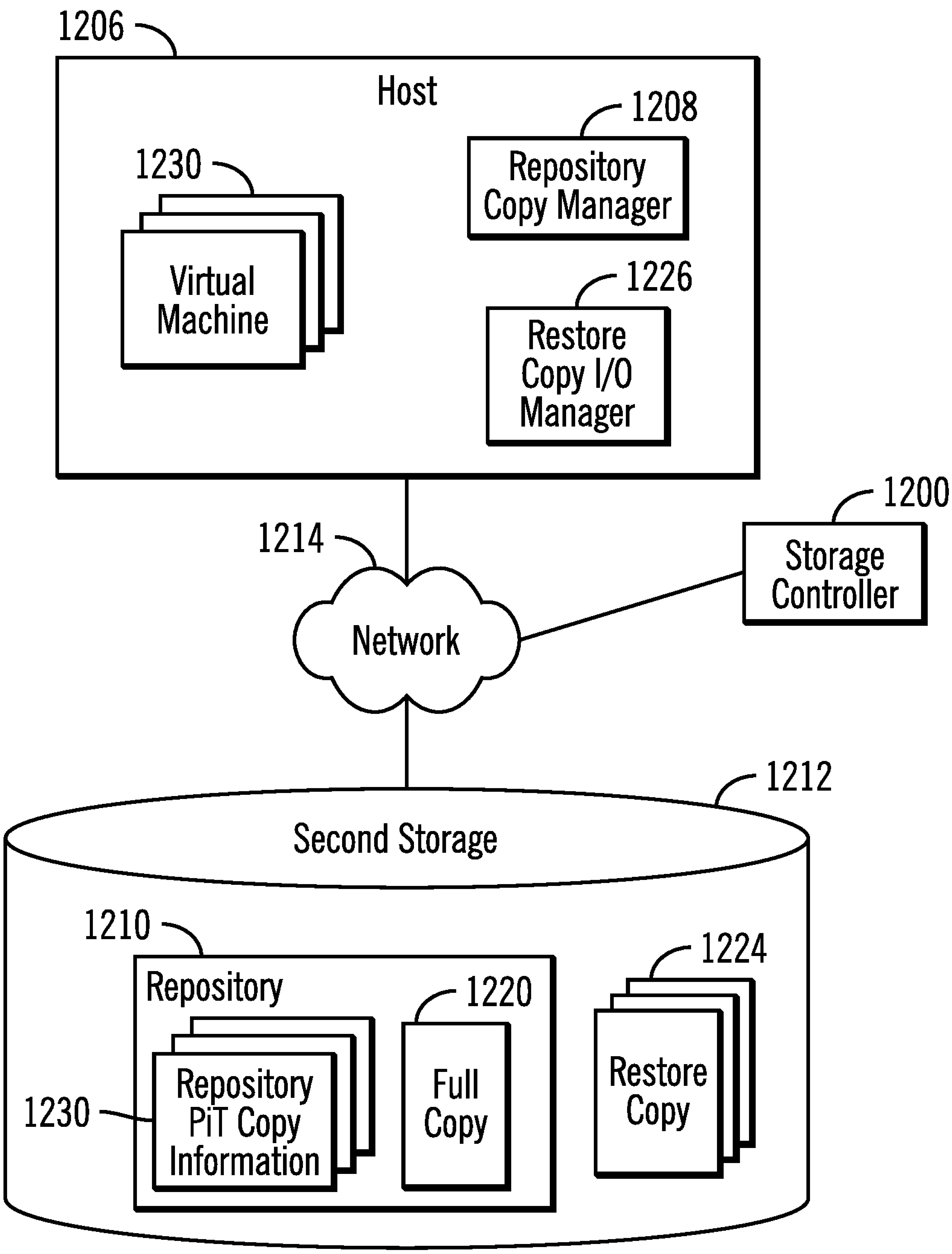
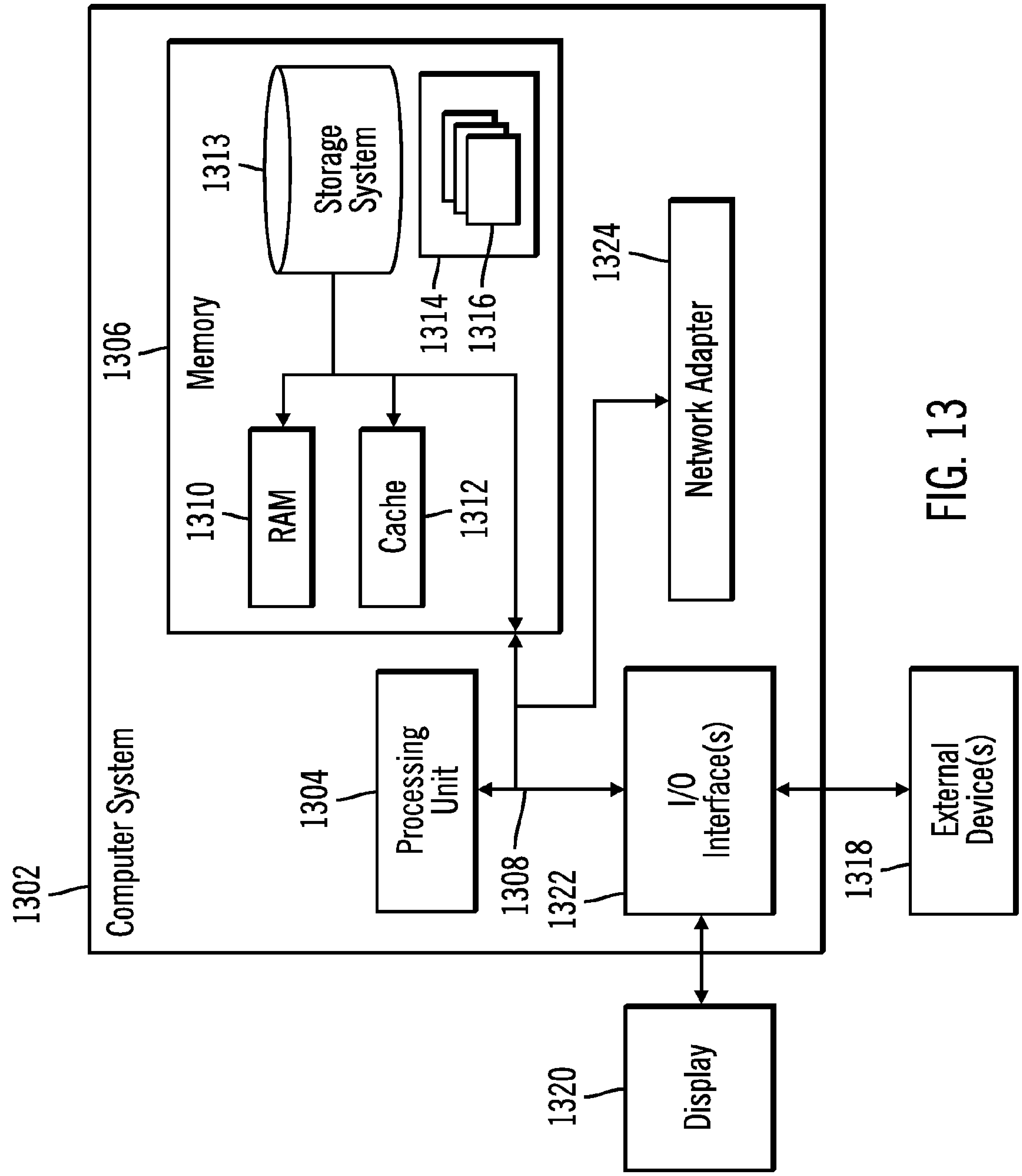

1206
Host
1208
Repository Copy Manager
1230
Virtual Machine
1226
Restore Copy I/O Manager
1214
Network
1200
Storage Controller
1212
Second Storage
1210
Repository
1220
Full Copy
1224
Restore Copy
1230
Repository PiT Copy Information

FIG. 12



1302
Computer System
1306
Memory
1313
Storage System
1314
1316
1310
RAM
Cache
1312
1324
Network Adapter
1304
Processing Unit
1308
1322
I/O Interface(s)
1318
External Device(s)
1320
Display

FIG. 13

CREATING A RESTORE COPY FROM A COPY OF SOURCE DATA IN A REPOSITORY HAVING SOURCE DATA AT DIFFERENT POINT-IN-TIMES AND READING DATA FROM THE REPOSITORY FOR THE RESTORE COPY

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a computer program product, system, and method for creating a restore copy from a copy of source data in a repository having source data at different point-in-times and reading data from the repository for the restore copy.

2. Description of the Related Art

In a storage environment, a storage controller may create point-in-time ("PiT") copies of a production volume using point-in-time copy techniques, such as the IBM FlashCopy® (FlashCopy is a registered trademark of IBM), snapshot, etc. A point-in-time copy replicates data in a manner that appears instantaneous and allows a host to continue accessing the source volume while actual data transfers to the copy volume are deferred to a later time. The PiT copy appears instantaneous because complete is returned to the copy operation in response to generating the relationship data structures without copying the data from the source to the target volumes. PiT copy techniques typically defer the transfer of the data in the source volume at the time the PiT copy relationship was established to the copy target volume until a write operation is requested to that data block on the source volume. Data transfers may also proceed as a background copy process with minimal impact on system performance. The PiT copy relationships that are immediately established in response to the PiT copy command include a bitmap or other data structure indicating the location of blocks in the volume at either the source volume or the copy volume. The PiT copy comprises the combination of the data in the source volume and the data to be overwritten by the updates transferred to the target volume.

When an update to a block in the source volume involved in a PiT copy relationship is received, the copy of the track as of the point-in-time must be copied to side file or the target volume before the new data for the track is written to the source volume, overwriting the point-in-time copy of the data.

SUMMARY

Provided are a computer program product, system, and method for creating a restore copy from a copy of source data in a repository having source data at different point-in-times and reading data from the repository for the restore copy. The source data is copied as of an initial point-in-time to a repository. Point-in-time copies at different point-in-times of the source data are initiated following the initial point-in-time. In response to completing each of the point-in-time copies, transmitting to the repository change information for the point-in-time copy indicating changed data in the source data that changed between the point-in-time of the point-in-time copy and a subsequent point-in-time. For each point-in-time copy, copying changed source data comprising source data indicated in the change information for the point-in-time copy as changed to the repository. Receiving a restore request to restore the source data as of a restore point-in-time. The restore copy is returned to the restore request before the source data in the repository as of the restore point-in-time is copied to the restore copy.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an embodiment of a storage environment.

FIG. **2** illustrates an embodiment of storage controller point-in-time copy information.

FIG. **3** illustrates an embodiment of repository point-in-time copy information.

FIG. **4** illustrates an embodiment of operations to create a repository having source data at different point-in-times.

FIGS. **5***a*, **5***b*, and **5***c* illustrate examples of states of the source data and the repository at different point-in-times.

FIG. **6** illustrates an embodiment of operations to create a restore copy.

FIG. **7** illustrates an embodiment of restore copy information.

FIG. **8** illustrates an embodiment of data unit information in the restore copy information for each data unit in the restore copy.

FIG. **9** illustrates an embodiment of operations to generate restore copy information.

FIG. **10** illustrates an embodiment of operations to process a write request to a restore copy.

FIG. **11** illustrates an embodiment of operations to process a read request to the restore copy.

FIG. **12** illustrates an embodiment of a storage environment including virtual machines.

FIG. **13** illustrates a computing environment in which the components of FIGS. **1** and **12** may be implemented.

DETAILED DESCRIPTION

Described embodiments provide techniques for creating a repository to maintain a full copy of the source data and point-in-time copies of the source data to allow the source data to be restored from the repository at different point-in-times. Further, with described embodiments, the operations to maintain the point-in-time ("PiT") copy information in the repository and perform the restore operations may be performed by a program component separate from the storage controller logic creating the PiT copies from the source data, so that the repository is separately maintained, used, and managed from the storage controller. Yet further, with described embodiments, a restore copy may be created and data requests to the data units in the restore copy may be returned from the repository to return data units as of a restore point-in-time of the restore copy.

FIG. **1** illustrates an embodiment of a data storage environment having a storage controller **100** managing access to a first storage **102** that includes source data **104**, such as a production volume used by different host systems. A host **106** includes a repository copy manager program **108** to manage the copying of the source data **104** to a repository **110** in a second storage **112**. The storage controller **100**, host **106**, and second storage **112** may communicate over a network **114**.

The storage controller **100** includes a point-in-time ("PiT") copy manager **116** to create point-in-time copies of the source data **104**, e.g., FlashCopy, snapshot, etc. When creating a PiT copy, the PiT copy manager **116** generates PiT copy information **200** on the PiT copy created as of a point-in-time. The storage controller **100** further includes an operating system **118**, including the code and logic to manage Input/Output ("I/O") requests to the source data

104. The operating system 118 may configure the source data 104 in one or more volumes and data is stored in data units, such as tracks, logical block addresses (LBAs), extents, etc. The PiT copy manager 116 may be a copy service supplied with the operating system 118.

The storages 102 and 112 may store tracks in a Redundant Array of Independent Disks (RAID) configuration where strides of tracks are written across multiple storage devices comprising the storages 102 and 112. The storages 102 and 112 may each comprise one or more storage devices known in the art, such as interconnected storage devices, where the storage devices may comprise hard disk drives, solid state storage device (SSD) comprised of solid state electronics, such as a EEPROM (Electrically Erasable Programmable Read-Only Memory), flash memory, flash disk, Random Access Memory (RAM) drive, storage-class memory (SCM), etc., magnetic storage disk, optical disk, tape, etc.

The network 114 may comprise a network such as one or more interconnected Local Area Networks (LAN), Storage Area Networks (SAN), Wide Area Network (WAN), peer-to-peer network, wireless network, etc.

The PiT copy manager 116 performs a PiT copy operation that replicates data in a manner that appears instantaneous and allows a process to continue accessing the source volume while actual data transfers to the copy volume are deferred to a later time. The PiT copy appears instantaneous because complete is returned to the copy operation in response to generating the relationship data structures without copying the data.

The repository copy manager 108 copies changed data of the source data 104 at different point-in-times to changed data 120 in the repository 110, including an initial copy of all the source data, and copies the PiT copy information 200 at the storage controller 100 to the repository 110 to store as repository PiT copy information 300 in the repository 110. The repository copy manager 108 uses the PiT copy information 300 to determine changed data at the source data 104 to copy to the changed data 120. To copy all the source data as of an initial point-in-time to the repository 110, the repository copy manager 108 may modify the change information for the initial PiT copy, or PiT bitmap, having a bit for each data unit in the source data 104 to indicate that all the source data has changed to generate modified initial change information 200_0' that indicates that each data unit in the source data 104 has changed. The repository copy manager 108 may then copy all the source data 104 units in the modified initial change information 200_0' to the changed data 120 in the repository 110 to provide a full copy of the source data 104 as of the initial point-in-time (T_0) in the repository. The repository copy manager 108 would use the change information in PiT copy information 200 for PiT copies following the initial point-in-time to copy changed data at different point-in-times to the changed data 120 in the repository 110.

With described embodiments, the repository copy manager 108 maintains a separate full copy of the source data 104 and the PiT copy information 300 of the source data 104 in a separate storage 112 so that data can be restored to different point-in-times independently of the source data 104 in the first storage 102 and the PiT copy information 200*a* at the storage controller 100. Thus, the repository 110 by allowing for independent restore of the source data 104 provides a secure backup independent to protect from failure at the storage controller 100 or first storage 102.

In certain embodiments, the creation and management of the repository 110 is managed by a repository copy manager 108 that is separate from the storage controller 100 programs, such as the PiT copy manager 116 that creates the PiT copies.

The host 106 may further include a restore copy I/O manager 126 to manage I/O requests directed to restore copies 124 to ensure that data as of a restore time for which the restore copy 124 is create is returned from the restore copy 124 or the repository 110.

To generate the restore copy 124, the repository copy manager 108 may utilize restore copy information 130 indicating the source data in the changed data 120 that needs to be copied to the restore copy 124.

In the embodiment of FIG. 1, the repository creation and restore operations may be performed by program components, such as the repository copy manager 108 and the restore copy I/O manager 126, that are separate from the storage controller logic, e.g., the PiT copy manager 116 that created the PiT copies. In an alternative embodiment, the repository copy manager 108, the repository copy I/O manager 126, and/or the repository 110 may be maintained on the storage controller 100.

FIG. 2 illustrates an instance of the PiT copy information 200, for PiT copy at time T_i, which may comprise information maintained for the storage controller PiT copy information 200, and may include a PiT copy identifier 202 identifying the PiT copy created by the PiT copy manager 116 at the storage controller 100; a point-in-time 204 of the PiT copy 202, which may mean that data is consistent as of that point-in-time 204; change information 206 indicating which data or tracks in the source data 104 has changed since the point-in-time 204 and while the PiT copy 202 was open, which may comprise a bitmap having a bit for each data unit (e.g., track) that is set to one of two values indicating the data or track represented by the bit has or has not been updated since the point-in-time 204; and indicate the changed PiT data 208 comprising the data at the point-in-time 204 that was changed after the point-in-time 204 while the PiT copy 202 was open and still being updated. PiT copy information may be maintained for an initial PiT copy taken at the initial time T_0 and for subsequent PiT copies taken at subsequent point-in-times.

In one embodiment, the PiT copy may be completed or frozen at a time subsequent to the point-in-time 204, such that the changed PiT data 208 includes data changed from the point-in-time 204 until the PiT copy 202 was completed, e.g., frozen or a new PiT copy initiated, and does not include changed data after the PiT copy 202 was completed, e.g., frozen. A PiT copy 202 may be completed upon a freeze command or initiating a subsequent PiT copy at a subsequent point-in-time to the point-in-time 204. A completed PiT copy 202 may be consistent as of the point-in-time. Other techniques may be used to complete, e.g., freeze, the PiT copy.

FIG. 3 illustrates an embodiment of an instance of PiT copy information 300, maintained in the repository 110, that is generated from an instance of PiT copy information 200, from the storage controller 100, and includes a PiT copy identifier 302 identifying the PiT copy 202 created by the PiT copy manager 116 at the storage controller 100; a point-in-time 304 of the PiT copy 302, which may mean that data is consistent as of that point-in-time 304; change information 306 indicating which data or tracks in the source data 104 has changed since the point-in-time 304 and while the PiT copy 302 was open, which may comprise a bitmap; and a repository offset 308 indicating an offset in the changed data 120 at which the data indicated in the change information 306 for the PiT copy 302 is located. The location in the changed data **120** of a particular changed data unit for a PiT copy **302** can be determined at an offset from the repository offset **208** in the changed data **120**, such as by summing the repository offset **208** with the number of the changed unit indicated in the change information **306**, i.e., the jth data unit, times a length of the data unit. In this way, changed data from a PiT copy **300** are written sequentially after the last written data in the change data **120** in the repository **110**.

FIG. **4** illustrates an embodiment of operations performed by the repository copy manager **108** and the PiT copy manager **116** to copy the source data **104** to the repository **110** at different points-in-time from the PiT copies for the source data **104**, such as a production volume, created by the PiT copy manager **116**. Upon initiating (at block **400**) repository copy operations for source data **104**, the repository copy manager **108** sends (at block **402**) a command to the PiT copy manager **116** to create an initial PiT copy 200_0 at an initial point-in-time, referred to herein as T_0. The repository copy manager **108** may modify the initial change information 200_0 to generate modified initial change 200_0' indicating that all source data units in the source data have changed. The repository copy manager **108** may send (at block **406**) to the repository **110** the modified initial change information 200_0', which indicates that all data units have changed and the changed PiT data 208_0 for the initial PiT copy (T_0) indicated in the modified initial change information 200_0', which indicates all the source data as changed. All the source data as of the initial point-in-time is copied to the changed data **120** in the repository **110**. The PiT copy 200_0' from the storage controller **100** would be stored as PiT copy information 300_0 with a repository offset **308** of zero because the start of the data from the initial PiT copy T_0 is stored at the beginning of the changed data **120** in the repository **110**.

The repository copy manager **108** may send (at block **408**) a command to the PiT copy manager **116** to create a subsequent PiT copy (T_{i+1}) at a subsequent point-in-time T_{i+1}, which would result in PiT copy information $200_{(i+1)}$. In one embodiment, the repository copy manager **108** may send individual commands to the PiT copy manager **116** to create PiT copies and freeze a previously created PiT copy. In an alternative embodiment, the repository copy manager **108** may send one command to the PiT copy manager **116** to instruct it to periodically create PiT copies of the source data **104**, where the creation of a subsequent PiT copy freezes the previously created PiT copy. Still further, the PiT copy manager **116** at the storage controller **100** may be independently creating PiT copies without prompting from the repository copy manager **108**.

Upon determining (at block **410**) that the previous point-in-time copy at T_i completed, the repository copy manager **116** sends (at block **412**) to the repository **110** for storage the PiT copy information 200_i for the PiT copy at time T_i, including the change information 206_i and the changed PiT data 208_i to store in the repository PiT copy information 300_i for T_i. In one embodiment, a PiT copy at point-in-time T_i may be completed upon the creation of the subsequent PiT copy at time T_{i+1}. In an alternative embodiment, a PiT copy may be completed by issuing a freeze command to freeze the PiT copy so that the change information **206** stops indicating changes to the source data **104**. The repository copy manager **108** creates (at bock **414**) in the repository PiT copy information **300**, from the received PiT copy information 200_i.

The repository copy manager **108** may transmit (at block **416**) the source data **104** at the subsequent time (T_{i+1}) that has changed since the previous time to the changed data **120** in the repository **110** to make the full copy current as of the subsequent time T_{i+1}. To perform this operation, the repository copy manager **108** may determine from the change information **206** for the previous PiT copy at T_i the data units in the source data **104** that have changed and then copy those data units from the source data **104** to the changed data **120**. In this way, at the subsequent point-in-time T_{i+1} when the subsequent PiT copy (T_{i+1}) is created, data units in the source data **104** that have changed between the previous point-in-time T_i and the subsequent point-in-time T_{i+1} are copied to the changed data **120** in the repository **110**.

The repository copy manager **108** may indicate (at block **418**) in the repository offset **308** the data unit offset in the changed data **120** where the changed data for the PiT copy at T_i begins in the changed data **120**. If (at block **420**) further point-in-time copies are to be created, then the repository copy manager **108** considers (at block **422**) the current subsequent point-in-time copy at (T_{i+1}) to be the previous point-in-time copy at T_i during the next iteration of processing the next PiT copy. In this way, i is effectively incremented as the current subsequent point-in-time T_{i+1} becomes the previous point-in-time and a next point-in-time becomes the new subsequent point-in-time. Control then proceeds back to block **408** to create or process a new subsequent PiT copy (T_{i+i}) at a new subsequent point-in-time T_{i+1}. If (at block **420**) there are no further PiT copies to process, then control ends.

With the described embodiments of FIG. **4**, a repository **110** of a full copy of the source data **104** having PiT copy information for different PiT copies is created independently of the storage controller **100** operations to create the PiT copies. In this way, the logic or operations of the repository management operations are independent of the storage controller **100** logic to create PiT copies.

FIGS. **5***a*, **5***b*, **5***c* illustrate examples of the operations of FIG. **4** to copy all the source data at different point-in-times to the changed data **120** and repository PiT copy information **200***b*. FIG. **5***a* shows the state at an initial time T_0 of the source data 104_0 in the source storage **102** that is copied to the changed data 120_0 in the repository **110**, which may comprise all the source data **104** as of the initial time T_0. Further, a PiT information copy 200_0 is created at the initial time T_0 which has change information 206_0' indicating that all the source data has changed at the storage controller **100**.

FIG. **5***b* shows a time T_1 when a second PiT copy 200_1 at T_1 is created, which may result in the completion or freezing of the previous PiT copy 200_0 at T_0 having changed data 208_0 as of the point-in-time T_0 before changes occurring between T_0 and T_1. The repository copy manager **108** copies the changed data between T_0 and T_1, shown as units **520** and **522**, to the changed data 120_1 in the repository **110** to update the changed data 120_1 to have changed data **520**, **522** as of T_1, which follows the initial written data. Further, the completed, or frozen, PiT copy information 200_0 at T_0 is copied to the repository **110** to store as repository PiT copy 300_0, which includes an offset 308_0 indicating zero because the source data for the initial PiT copy 200_0 is the first data written to the changed data 120_1.

FIG. **5***c* shows a time T_2 when a third PiT copy 200_2 at T_2 is created, which may result in the completion or freezing of the PiT copy 200_1 at T_1 having changed data 208_1 as of the point-in-time T_1 before changes occurring between T_1 and T_2. The repository copy manager **108** copies the changed data between T_1 and T_2, shown as **524** and **526**, to the repository **110** to update the changed data 120_2 to have data as of T_2. Further, the completed, or frozen, PiT copy information 200_1 at T_1 is copied to the repository **110** to store as repository PiT copy 300_1, which includes a repository offset 308_1 indicating the data unit offset in the repository 120_2 at which the changed data **520** and **522** begins.

FIG. **6** illustrates an embodiment of operations performed by the repository copy manager **108** to create a restore copy **124** from the repository **110** as of a restore time (T_R), which may comprise a point-in-time of one of the repository point-in-time copies 200_b. Upon processing (at block **600**) a restore request to create a restore copy of the source data as of the restore time (T_R), the restore copy manager **108** configures (at block **602**) an empty restore copy **124** or restore volume. The restore copy manager **108** may further generate (at block **604**) restore copy information **130** for the restore copy having information on the changed data **120** that comprises the data for the restore3 copy **124** as of the restore point-in-time (T_R). The empty restore copy **124** is returned (at block **606**) to the restore request to make available for access immediately.

FIG. **7** illustrates an embodiment of restore copy information **700** generated at block **604** for a restore copy **124**, including a restore copy identifier (ID) **702** identifying the restore copy; a restore point-in-time (T_R) **704** the user selected to create the restore copy **124** to which the data from the changed data **120** for the restore is copied; a restore copy location **706** indicating a location in the second storage **112** at which the restore copy **702** is stored; and data unit information **708** providing information for each of the data units in the restore copy **702**.

FIG. **8** illustrates an embodiment of an instance of data unit information **708**, for data unit i indicating a data unit number **800** of the data unit i, which may comprise an integer number indicating a sequential number of the data unit in the source data **104**, e.g., a block number; a copy bit **802** indicating whether the data unit **800** is located in the restore copy **124** or only in the repository **110**; and a repository location **804** indicating a location, such as an offset, in the changed data **120** at which the data unit **800** is located in the repository **110**. The repository location **804** may comprise an absolute byte location in the changed data **120** or comprise an offset, e.g., block number, from which the byte location can be determined by multiplying the offset times a data unit length. The repository location **804** is used to locate the data unit as of the restore time (T_R) from multiple copies of the data unit in the changed data **120** at different point-in-times.

In alternative embodiments, the information used to locate the data units in the changed data **120** may be in different formats, such as indicating the PiT copy from which the data unit **800** is obtained or indicate other offset information that may be used to calculate the specific location in the changed data **120** of the data unit **800**.

FIG. **9** illustrates an embodiment of operations performed by the repository copy manager **108** to generate the restore copy information **700**, such as the operation performed at block **604** in FIG. **6**. Upon initiating the operation (at block **900**) to determine the restore copy information **700**, the repository copy manager **108** initializes (at block **902**) restore copy information **800**, such as by indicating the restore point-in-time **704**, the restore copy location **706**, and the data unit information **708**. The copy bit **802** for each data unit **800** may be set to indicate that the data unit is not yet located in the restore copy **124**. T_i is set (at block **904**) to T_R. The repository copy manager **106** determines (at block **906**) the changed data units **120** for the PiT copy 300_i at time T_i whose location has not been determined, i.e., do not have a value set for the repository location **804** for the data unit

For each determined changed data unit at step **906**, the repository copy manager **106** determines (at block **908**) from the repository offset **308**, for the PiT copy at Ti the location in the changed data **120** of the data unit. In one embodiment, the repository offset **308** may provide just information for the first changed data unit of the PiT copy and the repository copy manager **108** must determine the offset for a particular data unit based on the number of the changed data unit, i.e., jth changed data unit indicated in the PiT copy change information **306**, plus the repository offset 308_i, then that sum times a data unit length. Alternatively, the repository offset **308** may indicate the offset number in the changed data **120** for each changed data unit for PiT copies following the initial PiT copy at T_0.

For instance, in an embodiment, where the repository offset **308*i*** comprises an offset at which the changed data for the PiT copy at Ti begins in the changed data **120**, the determined location may comprise the changed data unit number, comprising the number in the order in which the data unit was written for the PiT copy at Ti, plus the repository offset **308*i*** for the PiT copy at Ti, with this sum multiplied by the data unit length. For instance, if the data unit is the third data unit for the PiT copy at Ti written to the changed data **120** and the data unit number offset at which the changed data for the PiT copy at Ti begins in the change data **120** is 100, and the data unit length is 16 bytes, then the location for the data unit in the changed data **120** comprises the sum of three times 100 multiplied by 16, e.g., (3+100)×16, which equals 4800*th* byte in the changed data **120**. For each determined changed data unit, the repository copy manager **108** indicates (at block **910**) the determined location for the data unit in the repository location **804** for the data unit.

If (at block **914**) T_i equals T_0, then all repository PiT copies **300** at or before the restore time (T_R) have been considered, and control ends. Otherwise, if T_i does not equal T_0, then T_i is set (at block **916**) to T_{i-1} to proceed back to block **906** to consider the next PiT copy 300_{i-1} immediately preceding the last considered PiT copy 300_i. With the operations of FIG. **9**, changed data units are determined from the repository PiT copies **300** from the PiT copy closest to the restore time in reverse temporal order. In this way, if multiple PiT copies provide changed data for the same source data unit, then the changed data that is indicated for the restore copy being created comes from the PiT copy providing changed data for that data unit that is closest in time to the restore time T_R so that the restore copy **124** is provided with changed data **120** from the repository **110** as of the restore time T_R.

FIG. **10** illustrates an embodiment of operations performed by the restore copy I/O manager **126** to process a write request to the restore copy **124**. Upon processing (at block **1000**) a write request to the restore copy, the received write data is applied (at block **1002**) to the restore copy **124**.

FIG. **11** illustrates an embodiment of operations performed by the restore copy I/O manager **126** to process a read request to a requested data unit in the restore copy **124**. Upon processing (at block **1100**) the read request to a data unit j in the restore copy **124**, if (at block **1102**) the copy bit 802_j for data unit j indicates that the requested data unit is in the restore copy **124**, then the data unit is accessed (at block **1104**) from the restore copy **124** and returned to the request. Otherwise, if the request copy **124** does not store the requested data unit as of the restore time (T_R), then the restore copy I/O manager **126** determines (at block **1106**) from the repository **110** the location in the repository **110** for the changed data for a point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time (e.g., indicated in the repository offset **308**$_j$ for the data unit j). The requested data unit j is accessed (at block **1108**) from the determined location in the changed data **120** in the repository **110** to return to the read request. The accessed data unit j may be copied (at block **1110**) to the restore copy **124** to make available for future read requests and the copy bit **802**$_j$ for data unit j is updated to indicate the data unit is in the restore copy **124**. In an alternative embodiment, the accessed data unit may not be copied to the restore copy **124**, and instead always returned from the changed data **120** in the repository **110**.

With the described embodiments, the restore copy **124** may be generated almost instantaneously because no data needs be copied to the restore copy **124** before it is made available for access. Upon processing read request, the restore copy information **700** may be used to determine whether requested read data from the restore copy **124** is at the restore copy **124** or only in the changed data **120** to determine whether to return the requested data from the restore copy **124** or the changed data **120**.

FIG. **12** illustrates an embodiment in which the host **1206** implements a plurality of virtual machines **1230** for which different restore copies **1224** are created for use by the different virtual machines **1230**. Elements **1200**, **1206**, **1208**, **1214**, **1226**, **1210**, **1212**, **1220**, **1224** and **1230** in FIG. **12** are the same as corresponding elements **100**, **106**, **114**, **108**, **126**, **110**, **112**, **120**, **124**, and **300** described with respect to FIG. **1**. However, in FIG. **12**, the repository copy manager **1208** creates restore copies **1224** for the different virtual machines **1230** to use and the restore copy I/O manager **1226** manages virtual machine **1230** read and write requests to their respective restore copies **1224** as discussed with respect to the restore copy I/O manager **126**. In an alternative embodiment, each virtual machine **1230** may include its own instantiation of the repository copy manager **1208** and the restore copy I/O manager **1226**. In this way, the different virtual machines **1230** may access their own separate restore copies **1224**, and receive data as of the restore point-in-time (T_R) from the repository PiT copy information **200***b* if needed.

The present invention may be a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a wave-guide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

Computer readable program instructions for carrying out operations of the present invention may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

These computer readable program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/

or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

The computational components of FIG. **1** and FIG. **12**, including the storage controller **100**, **1200** and host **106**, **1206** may be implemented in one or more computer systems, such as the computer system **1302** shown in FIG. **13**. Computer system/server **1302** may be described in the general context of computer system executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. Computer system/server **1302** may be practiced in distributed cloud computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed cloud computing environment, program modules may be located in both local and remote computer system storage media including memory storage devices.

As shown in FIG. **13**, the computer system/server **1302** is shown in the form of a general-purpose computing device. The components of computer system/server **1302** may include, but are not limited to, one or more processors or processing units **1304**, a system memory **1306**, and a bus **1308** that couples various system components including system memory **1306** to processor **1304**. Bus **1308** represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnects (PCI) bus.

Computer system/server **1302** typically includes a variety of computer system readable media. Such media may be any available media that is accessible by computer system/server **1302**, and it includes both volatile and non-volatile media, removable and non-removable media.

System memory **1306** can include computer system readable media in the form of volatile memory, such as random access memory (RAM) **1310** and/or cache memory **1312**. Computer system/server **1302** may further include other removable/non-removable, volatile/non-volatile computer system storage media. By way of example only, storage system **1313** can be provided for reading from and writing to a non-removable, non-volatile magnetic media (not shown and typically called a "hard drive"). Although not shown, a magnetic disk drive for reading from and writing to a removable, non-volatile magnetic disk (e.g., a "floppy disk"), and an optical disk drive for reading from or writing to a removable, non-volatile optical disk such as a CD-ROM, DVD-ROM or other optical media can be provided. In such instances, each can be connected to bus **1308** by one or more data media interfaces. As will be further depicted and described below, memory **1306** may include at least one program product having a set (e.g., at least one) of program modules that are configured to carry out the functions of embodiments of the invention.

Program/utility **1314**, having a set (at least one) of program modules **1316**, may be stored in memory **1306** by way of example, and not limitation, as well as an operating system, one or more application programs, other program modules, and program data. Each of the operating system, one or more application programs, other program modules, and program data or some combination thereof, may include an implementation of a networking environment. The components of the computer **1302** may be implemented as program modules **1316** which generally carry out the functions and/or methodologies of embodiments of the invention as described herein. The systems of FIGS. **1** and **12** may be implemented in one or more computer systems **1302**, where if they are implemented in multiple computer systems **1302**, then the computer systems may communicate over a network.

Computer system/server **1302** may also communicate with one or more external devices **1318** such as a keyboard, a pointing device, a display **1320**, etc.; one or more devices that enable a user to interact with computer system/server **1302**; and/or any devices (e.g., network card, modem, etc.) that enable computer system/server **1302** to communicate with one or more other computing devices. Such communication can occur via Input/Output (I/O) interfaces **1322**. Still yet, computer system/server **1302** can communicate with one or more networks such as a local area network (LAN), a general wide area network (WAN), and/or a public network (e.g., the Internet) via network adapter **1324**. As depicted, network adapter **1324** communicates with the other components of computer system/server **1302** via bus **1308**. It should be understood that although not shown, other hardware and/or software components could be used in conjunction with computer system/server **1302**. Examples, include, but are not limited to: microcode, device drivers, redundant processing units, external disk drive arrays, RAID systems, tape drives, and data archival storage systems, etc.

The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the present invention(s)" unless expressly specified otherwise.

The terms "including", "comprising", "having" and variations thereof mean "including but not limited to", unless expressly specified otherwise.

The enumerated listing of items does not imply that any or all of the items are mutually exclusive, unless expressly specified otherwise.

The terms "a", "an" and "the" mean "one or more", unless expressly specified otherwise.

Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more intermediaries.

A description of an embodiment with several components in communication with each other does not imply that all such components are required. On the contrary a variety of optional components are described to illustrate the wide variety of possible embodiments of the present invention.

Further, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described does not necessarily indicate a requirement that the steps be performed in that order. The steps of processes described herein may be performed in any order practical. Further, some steps may be performed simultaneously.

When a single device or article is described herein, it will be readily apparent that more than one device/article (whether or not they cooperate) may be used in place of a single device/article. Similarly, where more than one device or article is described herein (whether or not they cooperate), it will be readily apparent that a single device/article may be used in place of the more than one device or article or a different number of devices/articles may be used instead of the shown number of devices or programs. The functionality and/or the features of a device may be alternatively embodied by one or more other devices which are not explicitly described as having such functionality/features. Thus, other embodiments of the present invention need not include the device itself.

The illustrated operations of the figures show certain events occurring in a certain order. In alternative embodiments, certain operations may be performed in a different order, modified or removed. Moreover, steps may be added to the above described logic and still conform to the described embodiments. Further, operations described herein may occur sequentially or certain operations may be processed in parallel. Yet further, operations may be performed by a single processing unit or by distributed processing units.

The foregoing description of various embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims herein after appended.

What is claimed is:

1. A computer program product for maintaining source data in a repository, wherein the computer program product comprises a computer readable storage medium having program instructions embodied therewith, the program instructions executable by a processor to cause operations, the operations comprising:

copying all the source data as of an initial point-in-time to a repository;

initiating point-in-time copies at different point-in-times of the source data following the initial point-in-time;

in response to completing each of the point-in-time copies, transmitting to the repository, change information for the point-in-time copy indicating changed data in the source data that changed between a point-in-time of the point-in-time copy and a subsequent point-in-time;

for each point-in-time copy, copying changed source data comprising source data indicated in the change information for the point-in-time copy as changed to the repository;

receiving a restore request to restore the source data as of a restore point-in-time;

separately generating a restore copy from the repository for the restore request comprising a logical copy to include a representation of the source data as of the restore point-in-time, wherein the restore copy does not include the source data as of the restore point-in-time after being generated; and

returning the restore copy to the restore request before the source data in the repository as of the restore point-in-time is copied to the restore copy, wherein the returned restore copy, which does not have the source data as of the restore point-in-time, is made available for immediate read and write access to the restore copy; and

copying data from the source data in the repository, as of the initial point-in-time and the changed data for at least one point-in-time copy as of a point-in-time closest to the restore point- in-time, to the restore copy in response to read requests after the generated restore copy is returned and made available for immediate read and write access.

2. The computer program product of claim **1**, wherein the operations further comprise:

receiving a read request requesting data in the restore copy;

determining whether the restore copy includes the requested data; and

returning the requested data from the changed data for one of the point-in-time copies in the repository at a point-in-time from the restore point-in-time to the initial point-in-time in the repository in response to determining that the restore copy does not include the requested data.

3. The computer program product of claim **2**, wherein the operations further comprise:

returning the requested data from the restore copy in response to determining that the requested data is in the restore copy.

4. The computer program product of claim **2**, wherein when multiple point-in-time copies in the repository provide changed data for the requested data, the changed data from the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time is returned as the requested data.

5. The computer program product of claim **1**, wherein the point-in-time copies are generated by a point-in-time copy program and wherein a repository copy program executed separately from the point-in-time copy program performs the operations of copying to the repository the change information and the changed data at the subsequent point-in-time indicated in the change information as changed.

6. The computer program product of claim **2**, wherein the operations further comprising:

indicating for each point-in-time copy in the repository a repository offset for indicating an offset in the repository where the changed data for the point-in-time copy is written.

7. The computer program product of claim **6**, wherein the operations further comprise:

generating restore copy information indicating location information used to locate each data unit in the repository at which the source data for the data units in the repository are stored, wherein when multiple point-in-time copies provide changed data as of the restore point-in-time, the location information is determined from the repository offset for the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time, wherein returning the requested data from the repository comprises using the location information indicated in the restore copy information for the requested data to access the requested data from the repository.

8. The computer program product of claim **7**, wherein the generating the restore copy information comprises:

for each point-in-time copy at or before the restore point-in-time, starting from the point-in-time copy closest to the restore point-in-time and considering point-in-time copies in reverse time order, performing:

determining changed data units for the point-in-time copy that have not already had location information indicated in the restore copy information;

for each of the determined changed data units, using the repository offset for the point-in-time copy to determine a location in the repository at which the determined changed data units are stored; and

indicating in the restore copy information the determined location in the repository for each of the determined changed data units.

9. The computer program product of claim **1**, wherein the operations further comprise:

receiving writes directed to the restore copy; and

applying the writes to update the restore copy.

10. The computer program product of claim **1**, wherein multiple restore requests are received from multiple virtual machines to create restore copies at different point-in-times, wherein for each of the virtual machines, operations are performed comprising:

receiving a read request from a requesting virtual machine comprising one of the virtual machines, requesting data in a restore copy for the requesting virtual machine; and

returning the requested data from the changed data for one of the point-in-time copies in the repository to the requesting virtual machine.

11. A system for maintaining source data in a repository, comprising:

a processor; and

a computer readable storage medium including program instructions executed by the processor to perform operations, the operations comprising;

copying all the source data as of an initial point-in-time to a repository;

initiating point-in-time copies at different point-in-times of the source data following the initial point-in-time;

in response to completing each of the point-in-time copies, transmitting, to the repository, change information for the point-in-time copy indicating changed data in the source data that changed between a point-in-time of the point-in-time copy and a subsequent point-in-time;

for each point-in-time copy, copying changed source data comprising source data indicated in the change information for the point-in-time copy as changed to the repository;

receiving a restore request to restore the source data as of a restore point-in-time;

separately generating a restore copy from the repository comprising a logical copy to include a representation of the source data as of the restore point-in-time, wherein the restore copy does not include the source data as of the restore point-in-time after being generated;

returning the restore copy to the restore request before the source data in the repository as of the restore point-in-time is copied to the restore copy, wherein the returned restore copy, which does not have the source data as of the restore point-in-time, is made available for immediate read and write access to the restore copy; and

copying data from the source data in the repository, as of the initial point-in-time and the changed data for at least one point-in-time copy as of a point-in-time closest to the restore point-in-time, to the restore copy in response to read requests after the generated restore copy is returned and made available for immediate read and write access.

12. The system of claim **11**, wherein the operations further comprise:

receiving a read request requesting data in the restore copy;

determining whether the restore copy includes the requested data; and

returning the requested data from the changed data for one of the point-in-time copies in the repository at a point-in-time from the restore point-in-time to the initial point-in-time in the repository in response to determining that the restore copy does not include the requested data.

13. The system of claim **12**, wherein when multiple point-in-time copies in the repository provide changed data for the requested data, the changed data from the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time is returned as the requested data.

14. The system of claim **12**, wherein the operations further comprising:

indicating for each point-in-time copy in the repository a repository offset for indicating an offset in the repository where the changed data for the point-in-time copy is written.

15. The system of claim **14**, wherein the operations further comprise:

generating restore copy information indicating location information used to locate each data unit in the repository at which the source data for the data units in the repository are stored, wherein when multiple point-in-time copies provide changed data as of the restore point-in-time, the location information is determined from the repository offset for the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time, wherein returning the requested data from the repository comprises using the location information indicated in the restore copy information for the requested data to access the requested data from the repository.

16. The system of claim 15, wherein the generating the restore copy information comprises:

for each point-in-time copy at or before the restore point-in-time, starting from the point-in-time copy closest to the restore point-in-time and considering point-in-time copies in reverse time order, performing:

determining changed data units for the point-in-time copy that have not already had location information indicated in the restore copy information;

for each of the determined changed data units, using the repository offset for the point-in-time copy to determine a location in the repository at which the determined changed data units are stored; and

indicating in the restore copy information the determined location in the repository for each of the determined changed data units.

17. The system of claim 11, wherein multiple restore requests are received from multiple virtual machines to create restore copies at different point-in-times, wherein, for each of the virtual machines, operations are performed comprising:

receiving a read request from a requesting virtual machine, comprising one of the virtual machines, requesting data in a restore copy for the requesting virtual machine; and

returning the requested data from the changed data for one of the point-in-time copies in the repository to the requesting virtual machine.

18. A method, comprising;

copying source data as of an initial point-in-time to a repository;

initiating point-in-time copies at different point-in-times of the source data following the initial point-in-time;

in response to completing each of the point-in-time copies, transmitting, to the repository, change information for the point-in-time copy indicating changed data in the source data that changed between a point-in-time of the point-in-time copy and a subsequent point-in-time;

for each point-in-time copy, copying changed source data comprising source data indicated in the change information for the point-in-time copy as changed to the repository;

receiving a restore request to restore the source data as of a restore point-in-time;

separately generating a restore copy from the repository comprising a logical copy to include a representation of the source data as of the restore point-in-time, wherein the restore copy does not include the source data as of the restore point-in-time after being generated;

returning the restore copy to the restore request before the source data in the repository as of the restore point-in-time is copied to the restore copy, wherein the returned restore copy, which does not have the source data as of the restore point-in-time, is made available for immediate read and write access to the restore copy; and

copying data from the source data in the repository, as of the initial point-in-time and the changed data for at least one point-in-time copy as of a point-in-time closest to the restore point- in-time, to the restore copy in response to read requests after the generated restore copy is returned and made available for immediate read and write access.

19. The method of claim 18, further comprising:

receiving a read request requesting data in the restore copy;

determining whether the restore copy includes the requested data; and

returning the requested data from the changed data for one of the point-in-time copies in the repository at a point-in-time from the restore point-in-time to the initial point-in-time in the repository in response to determining that the restore copy does not include the requested data.

20. The method of claim 19, wherein when multiple point-in-time copies in the repository provide changed data for the requested data, the changed data from the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time is returned as the requested data.

21. The method of claim 19, further comprising:

indicating for each point-in-time copy in the repository a repository offset for indicating an offset in the repository where the changed data for the point-in-time copy is written.

22. The method of claim 21, further comprising:

generating restore copy information indicating location information used to locate each data unit in the repository at which the source data for the data units in the repository are stored, wherein when multiple point-in-time copies provide changed data as of the restore point-in-time, the location information is determined from the repository offset for the point-in-time copy having a point-in-time from the restore point-in-time to the initial point-in-time and closest in time to the restore point-in-time, wherein returning the requested data from the repository comprises using the location information indicated in the restore copy information for the requested data to access the requested data from the repository.

23. The method of claim 22, wherein the generating the restore copy information comprises:

for each point-in-time copy at or before the restore point-in-time, starting from the point-in-time copy closest to the restore point-in-time and considering point-in-time copies in reverse time order, performing:

determining changed data units for the point-in-time copy that have not already had location information indicated in the restore copy information;

for each of the determined changed data units, using the repository offset for the point-in-time copy to determine a location in the repository at which the determined changed data units are stored; and

indicating in the restore copy information the determined location in the repository for each of the determined changed data units.

24. The method of claim 18, wherein multiple restore requests are received from multiple virtual machines to create restore copies at different point-in-times, wherein for each of the virtual machines, operations are performed comprising:

receiving a read request from a requesting virtual machine, comprising one of the virtual machines, requesting data in a restore copy for the requesting virtual machine; and

returning the requested data from the changed data for one of the point-in-time copies in the repository to the requesting virtual machine.

* * * * *